



E P I P H A N Y

INSTALLATION GUIDE

3.4

January 1999

Copyright and Trademarks

Copyright © 1998-1999 by Epiphany Marketing Software, Inc. All rights reserved.

This document and the software it describes are furnished under license and may be used or copied only in accordance with such license. Except as permitted by such license, the contents of this document may not be disclosed to third parties, copied, or duplicated in any form, in whole or in part, without the prior written permission of Epiphany Marketing Software, Inc.

This document contains propriety and confidential information of Epiphany Marketing Software, Inc. The contents of this document are for informational use only, and the contents are subject to change without notice. Epiphany Marketing Software, Inc. assumes no responsibility or liability for any errors or inaccuracies that may appear in this document.

Unpublished rights reserved under the copyright Laws of the United States.

Clarity™, Relevance™, Momentum™, and Magnitude™ are trademarks of Epiphany Marketing Software, Inc. All other products or name brands are trademarks of their respective holders.

Printed in the USA

Restricted Rights Legend

Software and accompanying materials acquired with United States Federal Government funds or intended for use within or for any United States federal agency are provided with "Restricted Rights" as defined in DFARS 252.227-7013(c)(1(ii)) or FAR 52.227-19.

CONTENTS

INTRODUCTION	5
About Epiphany.....	5
About Release 3.4	5
About This Guide	6
How To Use This Guide	6
Before You Begin	7
 CHAPTER 1	
PREPARING TO INSTALL EPIPHANY SOFTWARE	9
Order of Installation	11
Installing and Configuring Your EpiCenter Host Computer.....	11
Installing and Configuring a Windows NT Server Host	12
Hardware Requirements for Windows NT Server	12
Windows NT Server 4.0 Installation	13
Disk-Volume Configuration	15
Network Connectivity	15
Installing and Configuring a Solaris Host	16
Hardware Requirements for Solaris	16
Solaris Installation	18
Configuration of Solaris Resources	18
Disk-Volume Configuration	22
Network Connectivity	22
Installing and Configuring Your Database Server.....	23
Installing and Configuring SQL Server	23
Installing SQL Server	23
Configuring SQL Server	24
Assigning Data Stores in SQL Server	27
Creating Databases in SQL Server	28
Configuring the Temporary Database	28

Installing and Configuring Oracle	29
Installing Oracle	29
Configuring Oracle	31
Creating Control Files	33
Assigning Data Stores in Oracle	34
Creating Users for EpiMart and EpiMeta Tables	35
Preparing Your Application Host	35
Installing Windows NT Server 4.0 on Your Application Host	36
Installing Internet Information Server, Version 4.0	36
Installing Microsoft Office Components	37
Installing and Configuring Connectivity Packages for Your Database Client	
Software	37
SQL Utilities	37
Oracle8 Client	37

CHAPTER 2

INSTALLING EPIPHANY SOFTWARE 39

Performing the Installation	41
Application Server	42
Remote Administration	44
Configuring Epiphany Software	45
Configuring the Epiphany Web-server Proxy	45
Setting Up NT Performance Monitoring for Extraction	46
Verifying Your Installation	47

CHAPTER 3

MIGRATING FROM RELEASE 3.2 OR 3.3 49

INTRODUCTION

ABOUT EPIPHANY

Epiphany is the leading provider of Relationship Management (ERM) applications. Epiphany provides complete solutions for managing customer relationships by providing Web-based access to relevant information that is captured by various departments within a business organization. Epiphany applications allow you to browse through this information, gain new insights, and explore relationships and trends that might otherwise remain hidden within your various databases.

With packaged applications from Epiphany, you can:

- Browse current data from throughout your organization
- Drill down to analyze specific situations in detail
- Act on that information to satisfy current customers and cultivate new ones

ABOUT RELEASE 3.4

Epiphany Release 3.4 provides the following features and enhancements:

- Support for larger data sets than in previous versions
- Support for the following database-server platforms:
 - Oracle8[®] on Solaris[®] 2.6
 - SQL Server[®] 7.0 on Windows NT Server[®] 4.0

For best performance with SQL Server 7.0, Epiphany recommends that you install the Enterprise Versions of both Windows NT Server and SQL Server.

ABOUT THIS GUIDE

This manual is intended for database administrators and consultants who install and configure EpiCenter data marts and Epiphany applications. The installation procedures that are documented in this guide take between two and three hours to complete.

If specific instructions do not appear for a particular configuration step or option, default values are acceptable.

HOW TO USE THIS GUIDE

Epiphany provides on-line versions of our manuals in PDF form, with a full-text search index that you can use to locate keywords of interest to you. You can open and view these manuals with Adobe Acrobat Reader. However, if you want to use the keyword index to search across Epiphany manuals, you must use Acrobat Reader with Search or Acrobat Exchange.

You can download the free Acrobat Reader with Search from the following Adobe Web site: <http://www.adobe.com/prodindex/acrobat/readstep.html>. After you register with Adobe, select Acrobat Reader with Search from the pop-up menu, along with your language and platform, then click download and follow the instructions provided by Adobe.

After you have installed Acrobat Reader with Search, you can follow these steps to perform a keyword search:

- Step 1:** Open the PDF file for an Epiphany manual in Acrobat Reader with Search.
- Step 2:** From the main menu, choose Tools, then Search, and then Query.
- Step 3:** In the Find Results Containing text box, enter the keywords that you want to search for in the Epiphany documentation set. You can include multiple keywords and wild cards (an asterisk for multiple characters and a question mark for a single character).

Step 4: Click Search.

The documents that contain text that matches your search query are listed in rank order in the Search Results window.

Step 5: Double-click a document in the Search Results list to see the first occurrence that it contains.

Step 6: Use the Search Next and Search Previous buttons in the toolbar to navigate to other occurrences.

For further instruction on using Acrobat Exchange or Acrobat Reader, refer to the on-line guides for Acrobat Reader, which are available from the Help menu.

BEFORE YOU BEGIN

Please follow these steps before you install Epiphany software:

Step 1: Read Chapter 1 of this guide to ensure that you understand the following topics:

- The considerations involved in preparing the host computer, the operating system (OS), the database server, and networking connectivity for your EpiCenter datamart
- The steps that are required to prepare the host computer for your Epiphany application software

Step 2: Make sure that you have the appropriate installation and configuration manuals on hand for the hardware and software products on which your Epiphany software depends. Epiphany suggests that you obtain the following manuals, depending upon the database-server option you select.

- *Net8 Getting Started* (for use with Windows NT Server clients connected to Oracle/Solaris datamarts)
- *Oracle8 Installation Guide for SunSPARC Solaris*
- *Oracle8 Reference*

Before You Begin

- *Solaris 2.6 Hardware, SMCC Hardware Platform Guide*
- *Solaris 2.6 SPARC Installation Instructions*
- *SQL Server Administration Guide* (for Windows NT Server)
- *Start Here: Basics and Installation, Microsoft Windows NT Server*

Step 3: Please install the hardware and software products that are mentioned in this guide according to manufacturer instructions.

Step 4: Please follow the configuration recommendations that are suggested in this guide.

CHAPTER 1

PREPARING TO INSTALL EPIPHANY SOFTWARE

The process of installing Epiphany software requires that you first install and configure hardware and software components that are supplied by other manufacturers. The success of your Epiphany application depends on the correct installation and proper configuration of these components.

This manual provides general installation guidelines and configuration recommendations for the products on which Epiphany software depends. For specific instructions about a particular product, please refer to the manufacturer's documentation. The configuration recommendations discussed in this manual apply to Epiphany enterprise-relationship-management (ERM) applications only.

The components that you must install and configure before installing Epiphany software include:

- A dedicated host computer for your EpiCenter datamart (strongly recommended) running one of the following operating systems:
 - Windows NT Server 4.0 (Epiphany recommends the Enterprise Edition) with Service Pack 3 and the Microsoft Exchange mail-client utility from Options Pack 4
 - Sparc Solaris 2.6
- RAID disk-volume support (optional but recommended)

Preparing to Install Epiphany Software

- A relational database server from the following list:
 - SQL Server 7.0 (Epiphany recommends the Enterprise Edition)
 - Oracle8, Release 8.0.5 for Solaris
- A host computer for your Epiphany application running Windows NT Server
- Client utilities for your application host, including:
 - Microsoft Internet Information Server (IIS), Version 4.0, and the Microsoft Exchange Mail utility (available with Option Pack 4 for Windows NT Server)
 - The appropriate connectivity and management package:
 - SQL Server Utilities with a datamart that resides on SQL Server
 - Oracle8 Client for Windows with a datamart that resides on Oracle

Note: A separate application host is required only when you install your EpiCenter datamart on a Solaris host. If you install your datamart and Epiphany application software on the same computer (running Windows NT Server), you must install the appropriate client utilities listed above on your datamart host.

ORDER OF INSTALLATION

Figure 1 illustrates the order in which you install and configure the components of your Epiphany application.

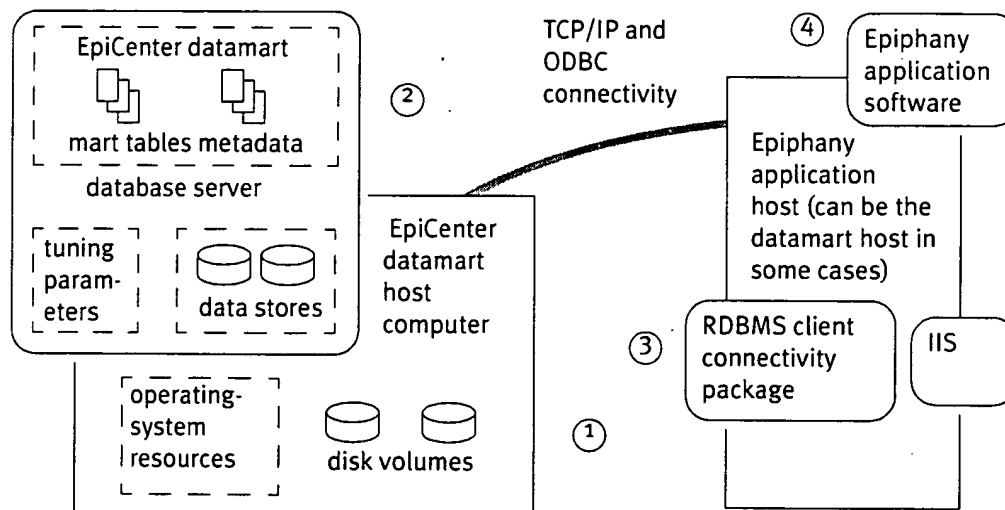


Figure 1: Epiphany Installation Components

INSTALLING AND CONFIGURING YOUR EPICENTER HOST COMPUTER

The specific installation and configuration tasks you must perform depend upon the host computer on which you intend to install your EpiCenter datamart. The sections that follow discuss the installation and configuration process for the following hosts:

- Pentium-based computers running Windows NT Server 4.0
- Sparc-based computers running Solaris 2.6

Installing and Configuring a Windows NT Server Host

This section provides guidelines for installing and configuring a computer running Windows NT Server 4.0 as an EpiCenter host. See “Installing and Configuring a Solaris Host” on page 16 for information about Solaris hosts.

Hardware Requirements for Windows NT Server

Please ensure that your computer meets the following minimum requirements:

- At least one CPU with a speed of 200 megahertz or higher (more CPUs are preferable), capable of supporting Windows NT Server 4.0

Refer to the Microsoft manual entitled *Hardware Compatibility List; Microsoft Windows NT*, Version 4.0, for listings of computers that support Windows NT Server. For enhanced performance, Epiphany recommends that you choose a computer that supports the Enterprise Edition of this OS.

- A minimum of 128 megabytes of random-access memory (RAM)
- Swap space of at least three times the size of RAM.
- The computer must be connected to a network and have TCP/IP network controller installed. Multiple network-card configurations are not currently supported.
- An adequate amount of disk space:

Epiphany software requires about 30 megabytes of disk space (including supporting software such as ODBC and JDBC drivers). The size of your data warehouse depends on the amount of data that you extract from your source systems. If you already have specifications for your datamart, you can use that information to estimate the size of your datamart data, taking into account the following considerations:

- The datamart includes two copies of the database and aggregates that are between three and ten times the base-table size, with additional space needed for staging tables, indexes, keys, and 100 megabytes of Epiphany metadata.
- Epiphany recommends that you reserve a separate log volume that is about 10 percent of the size of your datamart.

Installing and Configuring a Windows NT Server Host

- Epiphany recommends that you reserve a volume that is at least twice the size of your largest fact table for temporary tables.

Based on these considerations, Epiphany suggests that you reserve the following disk volumes for your EpiCenter datamart:

- a RAID-1+0 volume for your EpiMart data and EpiMeta metadata

Use the following formula to determine the size of this volume:

$$\text{mart_vol_size} = 2 * (\text{aggs} * \text{fact_rows} * (100 + \text{fact_row_width})) + 2 * (\text{dimension_rows} * (20 + \text{dimension_row_width})) + 400 \text{ Mb}$$

Replace *aggs* with a value between 3 and 10 depending on the degree to which your application requires aggregate data. Use a higher value for applications that require extensive use of aggregate data.

- a RAID-1 volume for the log device associated with your datamart

$$\text{log_vol_size} = (0.1 * \text{mart_vol_size}) + 10\text{MB}$$

- a RAID-5 volume for the temporary database

Use the following formula:

$$\text{temp_db_size} = \text{tmp_factor} * \text{fact_rows} * (100 + \text{fact_row_width})$$

Replace *tmp_factor* with a value between 2 and 10, depending on your application. Use a value of 2 if you plan to install the Magnitude option.

The Magnitude option requires two additional RAID-0 volumes for external sort space. Use the following formulas for these volumes:

$$\text{sort_working_vol} = \text{fact_rows} * (100 + \text{fact_row_size})$$

$$\text{sort_temp_vol} = 10 * \text{fact_rows} * (100 + \text{fact_row_size})$$

Windows NT Server 4.0 Installation

You must install Microsoft Windows NT Server 4.0, Service Pack 3.0 (or higher), and selected Microsoft Office components. For enhanced performance, Epiphany recommends that you install Enterprise Edition of Windows NT Server 4.0.

Installing and Configuring a Windows NT Server Host

Installing Windows NT Server

To install Windows NT Server 4.0, follow the directions provided in Part 2 of the Microsoft manual entitled *Start Here: Basics and Installation, Microsoft Windows NT Server*, Version 4.0, and follow these recommendations:

1. Select the default directory location in which to install Windows NT Server.
2. Assign a computer name of 15 or fewer characters and write this name down for future use.
3. Indicate the server type as a stand-alone server.
4. Enter your administrator account name and password.
5. Create a repair disk in case of an emergency.
6. Select the default list of components.

As part of the Windows NT Setup, after you finish setting up the network parameters, the Finishing Setup dialog box is displayed, which informs you that are about to start setting up Microsoft Internet Information Server, Version 2.0. Exit the installer at this time. You do not need to install IIS, Version 2.

Note: When installing the operating system, you might have to install additional drivers for 3rd-party disk drives, graphics cards, or other hardware.

Installing Service Pack 3.0 (or Higher)

Detailed instructions for installing your service pack are provided on the installation disk. Please follow those instructions to install the service pack.

Installing Microsoft Office Components

You can install Microsoft Office from the Microsoft Office CD. The installation program provides detailed instructions. Please follow those instructions to install the Microsoft Exchange messaging client, ODBC connection client, and any other components of Microsoft Office that you choose to include.

Disk-Volume Configuration

Take the following steps to configure disk volumes for your datamart:

- Step 1:** The disk volumes for your database server use NTFS format. Some computer models running Windows NT Server configure disks with FAT format by default. To convert disks from FAT to NTFS, use the Disk Administrator. From the Start Menu, choose Programs, then Administrative Tools, and then Disk Administrator.
- Step 2:** Refer to the disk-volume size figures that you calculated using the formulas in “Hardware Requirements for Windows NT Server” on page 12 for size and RAID-level information.
- Step 3:** If you are using a hardware RAID controller or RAID software, follow the manufacturer’s instructions for configuring your disk volumes. Otherwise, use the Disk Administrator to configure your disk volumes.
- Step 4:** In a command-window prompt, enter the following command to enable monitoring of disk I/O performance with the PerfMonitor utility:

```
diskperf -y
```

Network Connectivity

Epiphany software requires the TCP/IP network protocol. Please contact your network administrator for specific instructions regarding the installation and configuration of devices, drivers, and domains for this protocol.

When you have verified that you have network connectivity, you can proceed to the next stage of preparations. Refer to “Installing and Configuring SQL Server” on page 23 for further instructions.

Installing and Configuring a Solaris Host

This section provides guidelines for installing and configuring a computer running Solaris 2.6 as an EpiCenter host. This section discusses a number of topics involved in installing and configuring Solaris for use with Oracle8. For additional information, refer to the *Oracle8 Installation Guide for SunSPARC Solaris* and *Oracle Support Bulletin 104511.69*.

For information about a Windows NT Server host, see “Installing and Configuring a Windows NT Server Host” on page 12.

Hardware Requirements for Solaris

Please ensure that your computer meets the following minimum requirements:

- At least one CPU (two or more are preferred) with a speed of 200 megahertz or higher, capable of supporting Solaris 2.6
- At least 128 megabytes of RAM

For best performance, Epiphany suggests equipping your Solaris host with its full complement of RAM.

- Swap space of at least three times the size of RAM.
- A Solaris-supported CD-ROM drive that uses High Sierra or ISO 9660 format with the Rockridge extension
- The computer must be connected to a network and have the TCP/IP interface installed.
- An adequate amount of disk space:

The size of your data warehouse depends on the amount of data that you extract from your source systems. If you already have specifications for your datamart, you can use that information to estimate the size of your datamart data, taking into account the following considerations:

- The datamart includes two copies of the database and aggregates that are between three and ten times the base-table size, with additional space included for staging tables, indexes, keys, and 100 megabytes of Epiphany metadata.

Installing and Configuring a Solaris Host

Solaris Installation

Follow the directions provided with your Solaris software to install the operating system and current patch from Sun Microsystems.

Configuration of Solaris Resources

You must configure the following Solaris resources to support an Oracle database server for your EpiCenter datamart:

- Semaphores, shared-memory segments, and the paging threshold in the **/etc/system** file
- User **oracle** and group **dba**
- Automatic start-up and shut-down for Oracle
- Disk volumes
- Network Connectivity

The sections that follow describe these activities.

Updating the /etc/system File

As **root**, you must edit the **/etc/system** file to configure the following system resources for Oracle:

- semaphores
- shared-memory resources
- swapping threshold

- Epiphany recommends that you provide space for temporary tables that is at least twice the size of your largest fact table, and space for rollback segments that 10 percent of the size of your datamart.
- Epiphany recommends that you use separate devices for redo logs and archived log files.

Based on these considerations, Epiphany suggests that you configure a single RAID-1+0 array for your EpiCenter datamart, and that you use the following formula to calculate its size:

Use the following formula to determine the size of this volume:

```
data_size =      2 * (aggs * fact_rows * (100 + fact_row_width))
               + 2 * (dimension_rows * (20 + dimension_row_width)) + 400MB
               + (0.1 * mart_vol_size) + 10MB
               + tmp * fact_rows * (100 + fact_row_width)

rollback_seg_size =    mart_size / 5

mart_size = data_size + rollback_seg_size
```

Replace *aggs* with a value between 3 and 10, depending on the degree to which your application requires aggregate data. Use a higher value for applications that require extensive use of aggregate data. Replace *tmp* with a value between 3 and 10, depending on your application. Use a value of 3 if you plan to install the Magnitude option.

Epiphany also suggests that you configure a separate volume for redo logs, and an additional volume for log archives. These volumes need not be RAID volumes.

The Magnitude option requires two additional RAID-0 volumes for external sort space. Use the following formulas for these volumes:

```
sort_working_vol = fact_rows * (100 + fact_row_size)

sort_temp_vol = 10 * fact_rows * (100 + fact_row_size)
```

Installing and Configuring a Solaris Host

Modifying the Shell Initialization Files

Take the following steps to modify the shell initialization files for the **oracle** and **epichnl** user IDs.

Edit the **.profile** file in the **oracle** home directory to add the following lines. You can either type them in or copy and paste them from the Epiphany installation CD. After you place the installation CD in the CD-ROM drive, you can use a text editor to display the **/cdrom/unix/profile.sh** file.

```
umask 022
ORACLE_BASE=/opt/oracle # or some other pathname
ORACLE_SID=epi
ORACLE_HOME=/${ORACLE_BASE}/product/version
ORACLE_TERM=termtype
PATH=.:${ORACLE_HOME}/bin:${ORACLE_BASE}/local:/bin:/usr/bin:\
/usr/ccs/bin:/usr/openwin/bin:/usr/5bin:/usr/ucb
DISPLAY=remote_workstation:0.0 # remote installation only
LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:${ORACLE_HOME}/lib
TNS_ADMIN=${ORACLE_HOME}/network/admin
ORA_NLS33=${ORACLE_HOME}/ocommon/nls/admin/data
export ORACLE_BASE ORACLE_SID ORACLE_HOME ORACLE_TERM PATH \
DISPLAY LD_LIBRARY_PATH TNS_ADMIN ORA_NLS33
ORAENV_ASK=no
```

If you use the C shell, add the following lines the **.cshrc** file as well, which you can copy from the **/cdrom/unix/cshrc.csh** file of the installation CD.

```
umask 022
setenv ORACLE_BASE /opt/oracle # match pathname in .profile
setenv ORACLE_SID epi
setenv ORACLE_HOME ${ORACLE_BASE}/product/version
setenv ORACLE_TERM termtype
setenv PATH .:${ORACLE_HOME}/bin:${ORACLE_BASE}/local:/bin:\
/usr/bin:/usr/ccs/bin:/usr/openwin/bin:/usr/5bin:/usr/ucb
setenv DISPLAY remote_workstation:0.0 # remote installation only
setenv LD_LIBRARY_PATH ${LD_LIBRARY_PATH}:${ORACLE_HOME}/lib
setenv TNS_ADMIN ${ORACLE_HOME}/network/admin
setenv ORA_NLS33 ${ORACLE_HOME}/ocommon/nls/admin/data
set ORAENV_ASK=no
```

Increasing values for these parameters allocates more resources which effectively reduces the amount of physical memory that is available to processes. Add or modify the following lines in **/etc/system** to configure these resources:

```
set semsys:seminfo_semmni=70
set semsys:seminfo_semmns=100
set semsys:seminfo_semmns=200

set shmsys:shminfo_shmseg=10
set shmsys:shminfo_shmmni=100
set shmsys:shminfo_shmmni=1
set shmsys:shminfo_shmmmax=max_shared_size

set lotsfree=pages
```

Replace *max_shared_size* with a value that is between 80 percent and 90 percent of the size of physical memory in bytes. The value for **shmmax** indicates a total allowable size for shared memory; it does not allocate any memory resources.

Replace *pages* with the minimum number of pages that your computer requires in order to prevent unnecessary swapping. Epiphany recommends a value between 2 percent and 6 percent of the total number of pages in RAM. Use the following formula calculate the number of 8-kilobyte pages in RAM:

$$total_pages = megs * 128$$

Replace *megs* with the number of megabytes of RAM that are installed on your computer.

Use the **reboot** command to reboot your Solaris host so that these semaphore and shared-memory settings can take effect.

Creating User and Group IDs

When your operating system has finished rebooting, log in as **root**. Use **admintool** or the **useradd** utility to create the **oracle** and **epichnl** user IDs and the **dba** group. Set the default group ID for both **oracle** and **epichnl** to **dba**. Propagate this account and group information to the Network Information Service (NIS).

Installing and Configuring a Solaris Host

Replace *oracle_home_path* with the value of ORACLE_HOME as it appears in the **.profile** file you created in the previous section.

Step 2: Create links to the **dbora** file in the system-initialization run-level directories:

```
ln -s /etc/init.d/dbora /etc/rc0.d/K10dbora
ln -s /etc/init.d/dbora /etc/rc2.d/S99dbora
```

Step 3: Create a symbolic link for the **listener.ora** file:

```
ln -s /var/opt/oracle/listener.ora /etc/listener.ora
```

Disk-Volume Configuration

As **root**, take the following steps to configure disk volumes for your datamart:

Step 1: Refer to the disk-volume size figures that you calculated using the formulas in “Hardware Requirements for Solaris” on page 16 for size and RAID-level information.

Step 2: If you are using a hardware RAID controller or RAID software, follow the manufacturer’s instructions for configuring your disk volumes. Otherwise, use the standard Solaris utilities for configuring disk partitions.

Step 3: Create three mount points on these disk volumes for the datafiles from which your Oracle tablespaces can be constructed.

Network Connectivity

The following steps enable network access for the Oracle8 database server:

Step 1: Add the following entry to the **/etc/services** file for your Oracle database server:

```
listener      1521/tcp      #TNS Listener
```

Step 2: Log in as a user other than **root**. Then use the **ftp** command to verify that your Solaris host is connected to the network and can transfer data.

In each file, replace *termtype* with the appropriate value for your terminal or terminal emulator. Replace *remote_workstation* with the name of the workstation or monitor from which you intend to perform the Oracle installation.

- Step 5:** Log in as user **epichnl** and copy the **.profile** and **.cshrc** files that you just edited from the **oracle** home directory to the home directory for **epichnl**.
- Step 6:** Enable remote execution access for user **epichnl** by adding the following line to the **.rhosts** file in the **epichnl** home directory:

```
application_host      epichnl
```

Replace *application_host* with the hostname of the Windows NT Server host computer on which you intend to install your Epiphany application. Entries in the **.rhosts** file are case sensitive, so be sure that you enter the hostname exactly as it appears in the NT domain.

Setting Up Automatic Start-up and Shut-down for Oracle

- Step 1:** As **root**, set up the automatic start-up and shut-down script for Oracle, and another for the Oracle system identifier by entering the following command lines into the **/etc/init.d/dbora** file:

```
ORACLE_HOME=oracle_home_path
ORACLE_OWNER=oracle
if [ ! -f $ORACLE_HOME/bin/dbstart -o ! -d $ORACLE_HOME ]
then
    echo "oracle startup: cannot start"
    exit
fi
case "$1" in
  'start')
    su - $ORACLE_OWNER -c $ORACLE_HOME/bin/dbstart &
    ;;
  'stop')
    su - $ORACLE_OWNER -c $ORACLE_HOME/bin/dbshut &
    ;;
  esac
su - $ORACLE_OWNER -c "lsnrctl start"
```

Installing and Configuring SQL Server

5. Choose the default settings for Unicode collation.
6. Select TCP/IP Sockets in the Select Network Protocols dialog box.
7. Choose the automatic start-up option for both SQL Server and SQL Executive.
8. When asked for the SQL Execution Log on Account, enter your password.
(You must have administrator privileges.)
9. For the TCP/IP Socket Number, use the default port number.

Configuring SQL Server

Epiphany recommends that you take the following steps to configure SQL Server:

Step 1: Configure selected database-server parameters.

Step 2: Configure SMP concurrency.

Step 3: Disable default serialization.

Step 4: Configure TCP/IP sockets as the networking interface for client applications.

The following sections describe these steps.

Configuring Database-Server Parameters

To configure database-server parameters for use with EpiCenter, log on as user **sa** and start the SQL Enterprise Manager. Then take the following actions:

Step 1: Select Register Server from the menu. Specify **sa** as the login ID and leave the password field blank. By default, SQL Server sets up the **sa** account with a null password. You should change this password as soon as it is convenient for you to do so.

Step 2: Enter the machine name in the server box and choose Server, then Register Server.

When you have verified that you have network connectivity, you can proceed to the next stage of preparations. See “Installing and Configuring Oracle” on page 29 for further instructions.

INSTALLING AND CONFIGURING YOUR DATABASE SERVER

This section provides guidelines and recommendations for:

- Installing and configuring the database server for your EpiCenter datamart
- Assigning data stores for your EpiCenter data
- Creating databases for your EpiCenter datamart and metadata

Installing and Configuring SQL Server

This section provides guidelines and recommendations for installing and configuring SQL Server for use with an EpiCenter datamart.

Note: For enhanced performance, Epiphany recommends that you install the Enterprise Edition of Version 7.0. This Edition can be installed only on hosts that are running the Enterprise Edition of Windows NT Server.

Installing SQL Server

Note: Before you install SQL Server 7.0, you must install Internet Explorer 4.01, Service Pack 1, from the Install Prerequisites dialog box of the SQL Server installer.

To install SQL Server, follow the directions provided on the installation CD, along with the following recommendations.

1. Use the default path for the SQL Server installation path.
2. Use the default value for Master-device creation.
3. Choose the 850 Multilingual character set in the Select Character Set dialog box.
4. Choose the default Sort Order (dictionary order, case insensitive).

Step 3: In the Server Manager window, right-click the icon for the server you just installed in the Microsoft SQL Servers tree, then choose the Configure option. Select the Configuration tab in the Server Configuration/Options dialog box. Change the following values as indicated:

- Locks: Set the value to 20,000.
- Memory: Assign as much memory as your system configuration allows. For example, 100,000 2-kilobyte blocks equals 200 megabytes of memory. Consult the *SQL Server Administration Guide* for more information.

The value that is displayed is a theoretical maximum, *not* the maximum level that the machine will handle. If you set the memory value higher than a value your machine can run, SQL Server will not restart.

- Open objects: Set the value to 15,000.
- Procedure Cache: Set the value to between 5 and 15, depending on your application

Configuring SMP Concurrency Parameter for SQL Server 7.0

SMP Concurrency controls the number of threads that SQL Server releases to the operating system. The effect of this action is to limit the number of CPUs that SQL Server uses. On a uniprocessor computer, the optimal value is 1. On a symmetric multiprocessor (SMP) computer, the value you choose depends on whether or not the host is dedicated to SQL Server. If the host is dedicated to database-server operations, you can set SMP Concurrency to -1, which automatically allocates all available CPUs to SQL Server.

If the server is not dedicated, then you must allow applications other than SQL Server to have adequate CPU resources. In particular, if you install Epiphany applications on the same host as your datamart, you must allow enough resources for those applications to provide adequate response time to users. Epiphany suggests that you set SMP concurrency to one or two less than the number of CPUs for computers that host both your datamart and your Epiphany application.

This Page Blank (uspto)

Configuring TCP/IP Sockets

Epiphany applications use the TCP/IP protocol to communicate with SQL Server. Take the following steps to configure set up TCP/IP sockets as the networking interface for SQL Server:

- Step 1:** From the Start Menu, choose Programs, then Microsoft SQL Server, and then SQL Client Configuration Utility.
- Step 2:** In the Net Library tab, select TCP/IP sockets from the Default Network pop-up menu, then click Done.

Assigning Data Stores in SQL Server

Use the SQL Server Manager to set up new database devices and new databases in your EpiCenter datamart. You need to set up the following database devices:

1. A device for your datamart data
2. A log device for your datamart data
3. A device for Epiphany metadata
4. A log device for metadata
5. A temporary-database expansion device
6. A log device for the temporary database

Note: Epiphany recommends that you always use different devices for data and logs. The naming scheme you choose for your devices should distinguish between related data and log devices. For example:

- EMRT1.DAT and EMRTLOG1.DAT for datamart data and logs
- EMTA1.DAT and EMTALOG1.DAT for metadata and logs
- ETMP1.DAT and ETMPLOG1.DAT for the expanded temporary database

Take the following steps to set up each database device:

- Step 1:** Choose your server the SQL Server Manager window, then right-click on Database Devices.
- Step 2:** Choose New Device from the pop-up menu.

Installing and Configuring SQL Server

The default value for SMP concurrency is 0 on an SMP machine, in which case SQL Server automatically reserves one less than the number of CPUs on board. On a uniprocessor machine, SMP concurrency is set to 1. If you choose Dedicated SMP Support, SMP concurrency is set to -1.

To set SMP concurrency, follow these steps:

- Step 1:** From the Enterprise Manager, select the name of your server and right-click Configure.
- Step 2:** In the Configuration tab, set the Show Advanced Options value to 1.
- Step 3:** When the advanced parameters appear in the table of configuration parameters, set the appropriate value of the SMP Concurrency parameter for your host.

Disabling Default Serialization

The Epiphany Application Server makes several SELECT INTO type queries to calculate results. In the default SQL SERVER configuration, these types of queries attempt to serialize themselves by blocking on a particular table. To avoid this problem, you should set the -T5302 flag. Follow these steps:

- Step 1:** In the Enterprise Manager window, right-click the Server.
- Step 2:** Select Configure.
- Step 3:** Click Parameters.
- Step 4:** Add -T5302.

For more information about this flag, consult the Microsoft knowledge-base-article number Q153441: "*SELECT INTO Locking Behavior.*"

To obtain access the Microsoft knowledge base, visit the Microsoft support site at <http://www.microsoft.com/support> and follow the directions to view on-line documentation.

This completes the configuration instructions for SQL Server and the preparations required for a datamart host running Windows NT Server. For information on preparing your Epiphany application host, refer to “Preparing Your Application Host” on page 35.

Installing and Configuring Oracle

This section provides guidelines and recommendations for installing and configuring Oracle8 for use with an EpiCenter datamart.

Installing Oracle

Follow the directions provided in the *Oracle8 Installation Guide* to install Oracle. The following steps summarize this process:

Step 1: Log in as user **oracle**, **cd** to **/cdrom/oracle805/orainst**, and enter one of the following commands:

```
./orainst /c      # for character-based installation
./orainst /m      # for Motif-based installation
```

Step 2: Select Default Install, then Install, Upgrade, or Deinstall Software, and then Install New Product with Database Objects.

Step 3: Confirm the pathnames of the ORACLE_BASE and ORACLE_HOME directories.

Step 4: Confirm the location of the installation-log files.

Step 5: Choose Install from CD-ROM in the Install Source dialog box.

Step 6: Confirm the name of the Oracle instance you wish to create. The instance name is **epi**, the value you set previously for the ORACLE_SID environment variable.

Step 7: Install all Oracle and Solaris on-line documentation and help. Make sure that the pathname of the documentation directory is correct in the ORACLE_DOC dialog box. The default pathname is acceptable.

Installing and Configuring SQL Server

Step 3: Enter the name for your device.

Step 4: In the Location pop-up menu, select the disk volume that you created for the device you are creating (datamart, datamart log, metadata, and so on) and assign the amount of space on that disk volume that you intend to allocate to the device. (Refer to “Hardware Requirements for Windows NT Server” on page 12 for information about the size calculations for disk volumes and data stores.)

Step 5: Click Create Now to create the device.

Creating Databases in SQL Server

After creating the new devices, you need to set up the databases for your datamart:

Step 1: Right-click the Database directory in the SQL Server Manager window.

Step 2: Select New Database from the pop-up menu. The New Database dialog box) is displayed.

Step 3: Enter the name of a database: either `epimeta` or `epimart`.

Step 4: Select the device you just created for your datamart.

Step 5: Select the log device for this database.

Step 6: Click Create Now to create the database.

Create the second database by following the same steps.

Configuring the Temporary Database

The default size of the temporary database, **tempdb**, is 2 megabytes. The default location for **tempdb** is in the master device for SQL Server. Epiphany recommends that you expand this database and place it in a separate device. This process involves placing **tempdb** in RAM, then moving it to the intended device. For information on moving and expanding the temporary database, please refer to the SQL Server documentation and Microsoft knowledge-base-article number Q187824: “*How to Move TEMPDB to a Different Device.*”

This Page Blank (uspto)

Installing and Configuring Oracle

Step 8: Select the following products to install:

- Oracle8 Enterprise
- SQL*Plus
- PL/SQL
- Net8
- Net8 Protocol Adapters
- Advanced Networking
- Solaris Documentation
- Oracle Solaris Installer
- On Line Text Viewer

Step 9: Choose Yes when you are asked to confirm default database start-up.

Step 10: Enter three pathnames for mount points as requested. If you have reserved separate disk volumes for Oracle data, use the pathnames to the root directory of each volume. Otherwise, use pathnames to directories within the disk volume you reserved for Oracle.

Step 11: Enter a suitable pathname for Oracle documentation when prompted. Epiphany suggest entering a pathname of the form:

oracle_home/doc

Replace *oracle_home* with the pathname of the home directory for your Oracle instance.

Note: At this point, the Oracle installer begins to download files. This process typically takes less than an hour, and requires periodic checking for status or error messages.

Step 12: When the installer displays the Information dialog box, choose OK in it and all subsequent dialog boxes, then exit the installer.

Installing and Configuring Oracle

Step 4: Check to see that the Oracle listener process is running by entering the following command:

```
lsnrctl status
```

If this process is not running, lsnrctl displays a number of “unable to connect” and “no listener” messages, among others. To start the listener process, enter the following command:

```
lsnrctl start
```

Step 5: Verify that there is an entry for your EpiCenter database in the `/var/opt/oracle/oratab` file. The entry should take the form:

```
SID:ORACLE_HOME: Y
```

Replace `SID` with the instance ID of your Oracle instance. Replace `ORACLE_HOME` with the pathname listed in the `ORACLE_HOME` environment variable.

Step 6: Make sure that the Oracle command file has correct permissions, as follows:

```
cd $ORACLE_HOME/bin
chmod 4751 oracle
ls -lg oracle
```

The permissions should be:

```
-rwsr-s--x  oracle  dba  oracle
```

Step 7: Log in as **oracle** to perform the remaining configuration tasks.

Step 8: Download the Epiphany configuration scripts for Oracle, as follows:

- a) Insert the Epiphany installation disk in the CD-ROM drive on your Solaris host.
- b) Enter the following commands:

```
cd $ORACLE_HOME/rdbms/admin
cp /cdrom/unix/*.{sh ,sql,ora} .
```

Replace *SID* with the *SID* for your Oracle instance.

Configuring Oracle

Epiphany suggests that you take the following steps to configure Oracle8. For detailed configuration instructions, refer to the chapter entitled “Configuring the Oracle8 System” in the *Oracle8 Installation Guide*.

Step 1: Log in as **root** and enter one of following shell commands to clear environment variables that might adversely affect the configuration process:

```
unset SRCHOME TMPDIR TWOTASK      # Bourne or Korn shell
unsetenv SRCHOME TMPDIR TWOTASK   # C shell
```

Step 2: Review the **root.sh** script, and if it is correct, run it. If not, you can update this script and then run it without having to rerun the installer:

```
cd $ORACLE_HOME/orainst
sh ./root.sh
```

Answer Y to the following prompt if it appears:

```
ORACLE_HOME does not match the home directory for Oracle.
OK to continue? [N]:
```

Step 3: Ensure that the Oracle instance is running:

```
/bin/ps -ef | grep pmon
```

If the instance is running, the **ps** command displays a process listing that includes the SID for your Oracle instance. If your instance is not running, enter the following commands to start it:

```
svrmgrl # starts the server manager
connect internal
startup
disconnect
exit
```

Installing and Configuring Oracle

Assigning Data Stores in Oracle

Epiphany provides a UNIX shell script, called **make_tablespaces.sh**, that you can use as an aid in configuring tablespaces for your datamart. Based on your input, this script produces an SQL script that contains the appropriate CREATE TABLESPACE commands for your datamart. If you have not already done so, follow the instruction in Step 8 of the Configuring Oracle section to download this script.

Use the **make_tablespaces.sh** shell script to specify the number of datafiles for the following tablespaces:

- Large fact tables
- Indexes on large tables
- Dimension tables
- Indexes on dimension tables
- Metadata tables
- Transient tables
- Other application-specific tables

Take the following steps to use this script:

Step 1: Run the **make_tablespaces.sh** script by entering the command:

```
$ORACLE_HOME/rdbms/admin/make_tablespaces.sh
```

Step 2: Review the **make_tablespaces.sql** script to ensure that the sizes and pathnames for datafiles are correct.

Step 3: Enter the following commands to execute the SQL script:

```
sqlplus oracle  
-- enter password  
@maketablespace.sql
```

Note: This SQL script creates tablespaces with specific names that EpiManager and EpiChannel recognize by default. If you use alternate names for tablespaces, you must define values for the macros that EpiChannel uses to locate those tablespaces. Refer to the *Epiphany System Guide* for details on Epiphany macros.

Step 9: Update the `$ORACLE_HOME/dbs/initSID.ora` file for your Oracle instance by creating a link to the **epi.ora** sample file, which you downloaded in the previous step:

```
cd $ORACLE_HOME/dbs
mv initSID.ora initSID.orig
ln -s ../admin/rdbms/admin/epi.ora initSID.ora
```

Step 10: Edit your **initSID.ora** file to provide values that are consistent with the size of your application. Please refer to the *Oracle8 Reference* for detailed information about specific initialization parameters and values.

Note: Epiphany applications typically perform large decision-support queries, as opposed to the large numbers of concurrent-but-brief queries of a typical on-line-transaction-processing (OLTP) application. Epiphany recommends that you configure Oracle using initialization parameters and values that are geared toward high TPC-D (Transaction Processing Performance Council benchmark D) performance. The sample file provides suggested values only, which Epiphany suggests that you edit to suit your application, and that you adjust over time as you learn more about performance in your specific circumstances.

Creating Control Files

Epiphany provides a sample SQL script, called **epi_rbctl.sql**, that you can edit and run to create to control files and perform other initialization tasks. As you edit this script, please be aware that quoted strings and pathnames are case sensitive in Oracle SQL. By convention, Epiphany tablespaces and datafiles use uppercase names. After you have edited the script, enter the following commands to run it using **svrmgrl**:

```
svrmgrl # start svrmgrl
connect internal;
@epi_rbctl.sql
exit
```

Note: The **epi_rbctl.sql** script calls several system-catalog-creation scripts that produce copious status messages and a number of “drop object” warnings that you can safely ignore.

Installing Windows NT Server 4.0 on Your Application Host

Installing Windows NT Server 4.0 on Your Application Host

If you plan to install your Epiphany application on the same host as your datamart, you do not need to install Windows NT Server again. Otherwise, please see, “Windows NT Server 4.0 Installation” on page 13, and “Network Connectivity” on page 15 for instructions on installing this operating system and configuring it.

Installing Internet Information Server, Version 4.0

Epiphany requires that you install Microsoft Internet Information Server (IIS) 4.0. IIS 4.0 is distributed with Windows NT Server, Options Pack 4.0. Detailed instructions for installing components of this options pack are provided on the installation disk. You can install additional components of the options pack on your application host if you choose.

Follow these recommendations when setting up IIS:

- Use the default values for the Options dialog box.
- You may specify different values in the Database Publishing Directory dialog box. To use the security features of IIS, be sure that the **wwwroot** directory resides on an NTFS file system.
- Select the Microsoft SQL Server driver to install.
- Set the Time Zone.
- In the Windows NT Server Services dialog, set the following services to Manual start-up: Certificate Authority and Content Index.

After installing the package, please reboot Windows NT Server.

For further information about IIS 4.0, please refer to the Microsoft support site, <http://www.microsoft.com/support>.

Creating Users for EpiMart and EpiMeta Tables

Epiphany provides a sample SQL script, called **epi_user.sql**, that you can use to create standard users (owners) for EpiCenter datamart and metadata tables. Edit this sample script, which you downloaded to **\$ORACLE_HOME/bin** in Step 8 of the previous section, and then use **sqlplus** to run it. Please be aware that user names in Oracle must be all uppercase.

```
sqlplus internal # start sqlplus
@epi_user.sql
exit
```

This completes the configuration instructions for Oracle and the preparations required for the datamart host. For information on preparing your Epiphany application host, please proceed to the next section.

PREPARING YOUR APPLICATION HOST

Epiphany applications require a minimum of 64 megabytes of RAM on your application host. You must install the following software packages on your application host before you install Epiphany software:

- Windows NT Server 4.0
- Microsoft Internet Information Server (IIS), Version 4.0
- Selected components of Microsoft Office

If your application host is not the same computer as your datamart host you must also:

- Install the appropriate client software for the database server on which your datamart resides.
- Configure connectivity for your client software.

The following sections describe these actions.

Installing and Configuring Connectivity Packages for Your Database Client Software

The Oracle installer installs Net8 as part of the client-utilities package. You use Net8 to establish connectivity with your datamart. To establish a connection to the Oracle instance on which the datamart resides, you must provide Net8 with the following information about that instance. You can use the Net8 Easy Config utility that comes with Oracle8 Client, or you can update the **tnsnames.ora** file directly. Either way, you must supply the following information:

- The protocol, in this case TCP/IP
- The hostname or IP address of the datamart host
- The port number of the TNS listener service on the datamart host
- The Oracle instance name (SID)

The **tnsnames.ora** file is located in the **Orant\Net80\Admin** folder in your Oracle installation directory. A typical entry for an Epiphany datamart takes the following form:

```
EPI=
  (DESCRIPTION=
    (ADDRESS=
      (PROTOCOL = TCP)
      (HOST = epihost)
      (Port = 1521)
    )
    (CONNECT_DATA = (SID = EPI)
  )
)
```

For additional details on configuring Net8, refer to the *Net8 Getting Started* manual.

This completes the tasks that you must perform to prepare your Epiphany application host. Please turn to Chapter 2 for instructions on installing Epiphany software.

Installing Microsoft Office Components

You can install Microsoft Office from the Microsoft Office CD. The installation program provides detailed instructions. Please follow those instructions to install the Microsoft Exchange mail client, ODBC connection client, and any other components of Microsoft Office that you choose to include.

Installing and Configuring Connectivity Packages for Your Database Client Software

If you plan to install Epiphany software on the same computer as your datamart, you do not need to install or configure additional client software. You can instead turn to Chapter 2, “Installing Epiphany Software.” If you plan to install your Epiphany application on a separate computer, please proceed with the sections that follow.

SQL Utilities

If you are using SQL Server for your datamart, you must install the SQL Utilities, option on the SQL Server installation CD, which provides appropriate instructions. After you have installed SQL Utilities, follow the steps in “Configuring TCP/IP Sockets” on page 27 to enable TCP/IP connectivity.

Oracle8 Client

If your datamart resides on an Oracle database server, you must install the Oracle8 Client package for Windows NT Server. Please observe the following recommendations:

- In the Select Installation Options dialog, choose Oracle8 Client.
- In the Select Oracle8 Client Configuration dialog, choose Database Administrator.
- Epiphany suggests that you install the Oracle documentation set on your hard drive.

Installing Epiphany Software

- Microsoft Internet Information Server (IIS), Version 4.0

If IIS is not already installed on your application host, you must install it before you attempt to install Epiphany software. Please review the guidelines for installing IIS listed in “Installing Internet Information Server, Version 4.0” on page 36.

- Selected components of Microsoft Office

If the Microsoft Exchange Messaging client and the ODBC connection client are not yet installed on your application host, you must install it. Please review the guidelines for installing these components described in “Installing Microsoft Office Components” on page 37.

If your EpiCenter datamart resides on a Windows NT Server host, you can use the same computer for your datamart and your Epiphany application.

The Epiphany installer offers to install several third-party system components that are required to support your Epiphany applications if they are not already present on your system. These components include:

- ODBC and JDBC drivers
- Sun JRE
- Microsoft Internet Explorer 4.0 (IE), including the Microsoft Java virtual machine

Please install IE 4.0 if the installer asks you to do so. The installer prompts you to exit if the Java virtual machine is not present and you do not install it.

After you have installed Epiphany software on the on the application host, you can install individual components of the EpiCenter Enterprise Manager (also referred to as EpiManager™) administrative utility on other computers to provide easier access for specific administrative tasks.

To install the complete set of standard Epiphany software on your application host, choose the Application Server installation option. To install an additional copy of selected Epiphany utilities on a remote host (which can run Windows 95, Windows 98, or Windows NT Server 4.0), choose the Remote Administration option.

Figure 2 on page 41 illustrates the components that you install on the Epiphany application host and any remote-administration hosts that you might add.

INSTALLING EPIPHANY SOFTWARE

This chapter describes the procedures for installing and configuring Epiphany application software. The standard software installation includes:

- the EpiManager™ administrative utility for configuring and managing your datamart and applications.
- the AppServer™ application server, which coordinates user requests for data and reports, returns results in HTML format, and supports navigation between applications.
- the EpiChannel™ extraction utility for downloading data from source systems into your EpiCenter datamart.
- other Epiphany components and utilities.
- drivers and other third-party software components on which Epiphany software depends.

Note: Epiphany application software does not have to be installed on the same computer on which your EpiCenter datamart resides.

The computer from which you run the Epiphany Application Server (AppServer™) must provide:

- Windows NT Server 4.0 set to run in 256-colors mode or higher
- 64 megabytes of RAM for use by Epiphany software

Application Server

Step 3: Double-click the **Setup.exe** icon in the CD-ROM folder.

Step 4: If the Microsoft Java virtual machine is not installed on your system, you are requested to install it. Please choose Yes in response to this prompt.

Step 5: The System Configuration screen displays your system's configuration data (for your information).

Follow the instructions below for either the Application Server or Remote Administration installation options. If you are installing Epiphany software for the first time, choose the Application Server option.

Application Server

The following steps apply to the Application Server installation option.

Step 1: Choose the Application Server option.

A Services window shows which services will be stopped before the copy operation begins.

Step 2: Enter the instance name. This is the name of the service as it will appear in the Service Control Panel.

An instance name distinguishes among multiple installations of the Epiphany software on the same machine. Typically, you will install the software once on a machine and thus have one instance. You are asked if this instance will be the default instance. If so, the **Setup** program makes this instance the default Web server destination.

Step 3: Enter the machine name on which the Application Server runs.

By default, this is the full DNS name of the server. Because of your local network configuration, you may need to use a unique name; for example, the machine name without the domain name. Check with your local system network administrator if you have any questions.

Step 4: Accept the default TCP/IP port for the Application Server if there is only one instance of the Application Server.

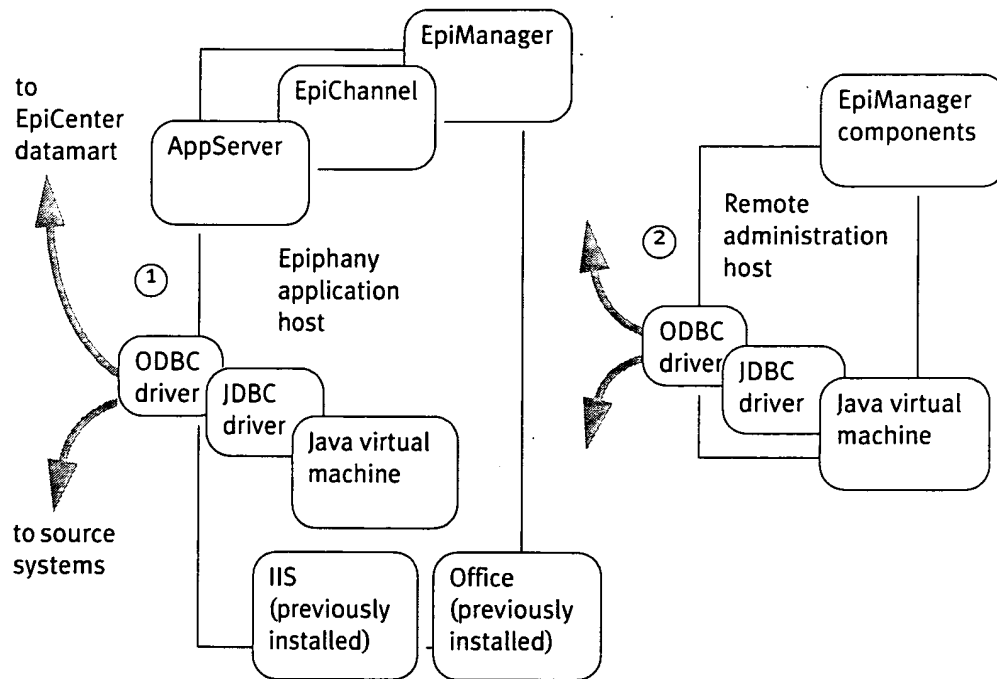


Figure 2: Epiphany Application Components

PERFORMING THE INSTALLATION

Take the following steps to install Epiphany software on your application host:

Step 1: Exit all open Windows applications.

If an AppServer instance is running, from the Start menu, go to **Settings\Control Panel\Services** and manually stop the Application Server. If you do not, the installer cannot properly download the DLLs that this release requires.

Step 2: Insert the Epiphany CD into your CD-ROM drive.

Remote Administration

Step 11: Select the components to be installed:

- Administrative Tools

You have the option of selecting individual components within the Administrative Tools option. If you are installing Epiphany software for the first time, choose the components that have been selected by default.

- Documentation

Epiphany provides technical manuals on line in PDF format with keyword-search indexes enabled. To use the PDF full-text index, you must have version 3.0 or higher of Acrobat Exchange or Acrobat Reader. You can download Acrobat Reader 3.0 from the Adobe Web site:

<http://www.adobe.com>.

Step 12: Select the default program-icon location.

Step 13: If this is an upgrade from Epiphany Release 3.2 or 3.3, and your datamart is already populated with data, refer to Chapter 3, "Migrating From Release 3.2 or 3.3" for further instructions. Otherwise, you can go to the Service Control Panel and start the service.

This completes the Application Server installation option.

Remote Administration

Step 1: Select the destination directory for the Epiphany software. The default is **C:\Program Files\Epiphany\instance_name**.

Step 2: Enter the location of the run-time reports and charts that end users will access. **C:\Program Files\Epiphany\instance_name\Charts** is the default. Create a new folder as requested.

Each instance must have a separate TCP/IP port. Check with your local system network administrator if this is the case.

Step 5: If your datamart resides on SQL Server, enter the name of the database server and the database name. The database name is the name of the EpiMeta database. This database does not need to exist as of yet. (The Registry keys are populated based on what you enter.)

If your datamart resides on an Oracle database server, enter the service name of the Net8 service associated with your datamart.

Step 6: If your datamart is on SQL Server, enter the user name and password for the database server. The user must have system administrator (SA) privileges on the EpiCenter host.

If your datamart resides on Oracle, enter the user name for your EpiMeta user (typically **EPIMETA**) and the password for that user.

Step 7: At this point, the installer asks if the instance name you entered in Step 1 is the default instance name. If you are installing Epiphany software for the first time, or if you plan to run a single instance of AppServer on this host, click Yes.

If you click No, your Web Server will not be set up to automatically route users to the Epiphany login page. In that case, users can access the login page by entering:

`http://machine_name/scripts/instance_name/Epiphany.dll`

Step 8: Select the destination directory for the Epiphany software. The default directory is:

`C:\Program Files\Epiphany\instance_name`

Step 9: Enter the location of the run-time reports and charts that end users will access. The default location is:

`C:\Program Files\Epiphany\instance_name\Charts`

Step 10: Create a new folder as requested.

Setting Up NT Performance Monitoring for Extraction

Step 2: Right-click your server icon and select Properties from the pop-up menu. The Properties dialog box is displayed in the Master Properties panel. Select WWW Services as the Master Properties. Click Edit. In the WWW Service Master Properties dialog box, select Directory Security. In the Anonymous Access and Authentication Control Panel, click Edit. The Authentication Methods dialog box is displayed. Select Allow Anonymous Access. Click Edit.

Step 3: Configure Anonymous Login with file access permissions for reading and writing to all Epiphany files. Enter the following attributes for the Anonymous user account:

- Username

A username that has file access permission for all of the Epiphany installed files and directories. This username also needs permission to read the Epiphany entries in the Registry.

Make sure that the directories that contain the **Epiphany.dll** and the **makechart.dll** files have execute permissions.

- Password

The password for this username.

Setting Up NT Performance Monitoring for Extraction

The **extract.exe** program is instrumented to use the standard NT Performance Monitoring facility, but you must take the following steps to enable the display of Epiphany data extraction processes in the NT Performance Monitor:

Step 1: Your **Epiphany\instance_name\win32** installation directory must contain these items: **EpiPerfMon.dll**, **EpiPerfMon.ini**, and **EpiPerfMon.reg**.

EpiPerfMon.reg has values specified for the following Registry key:

HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Epi\Performance

Step 3: Select the components to be installed:

- **Administrative Tools**

You have the option of selecting components of the following Administrative Tools: EpiCenter Manager, Web Builder, and Security Manager.

Although EpiCenter Manager includes Web Builder and Security Manager, these are available as separate programs. A person who does not need to have access to all of the features of EpiCenter Manager can use one of these components. The ticksheet-configuration features of EpiCenter Manager are available via Web Builder. Access rights and system permissions are available via Security Manager.

- **Documentation**

Step 4: Select the default program-icon location.

Step 5: If you are performing an upgrade from Epiphany Release 3.2 or 3.3 and your datamart is already populated with data, please refer to Chapter 3, “Migrating From Release 3.2 or 3.3” for further instructions. Otherwise, you can go to the Service Control Panel and start the service.

This completes the Remote Administration software-installation option.

CONFIGURING EPIPHANY SOFTWARE

After you have installed Epiphany software, you must configure the Epiphany Web-server proxy. In addition to configuring the proxy, Epiphany also recommends that you enable performance monitoring for data-extraction operations.

Configuring the Epiphany Web-server Proxy

The Epiphany Web-server proxy interacts with IIS 4.0. For the Epiphany Application Suite to function properly, set the IIS configuration values according to the instructions in this section.

Step 1: Open the IIS 4.0 Service Manager from the Start menu by choosing Programs, then Microsoft Internet Server, then Internet Service Manager.

Verifying Your Installation

The *Epiphany System Guide* gives instruction on how to use EpiManager to administer your Epiphany application. Please read Chapters 1 and 2 of that guide for background information before using EpiManager.

- Enter your e-mail password to receive notification of Epiphany system status. You will use the Configuration dialog box in EpiManager to set this up. Instructions for doing so appear in Chapter 3 of that guide.
- Run the Epiphany extraction program (EpiChannel) to extract data from your source systems and place them in your EpiCenter datamart. EpiChannel performs the extraction jobs that you defined in EpiCenter Manager.
- Start AppServer, the Epiphany Application Server. Refer to the *Epiphany System Guide* for details.

Running **EpiPerfMon.reg** enters this key and its values into the Registry. Before doing so, edit this file so that the value **Library** points to the correct path where **EpiPerfMon.dll** resides, that is, your current Win32 directory.

Edit **EpiPerfMon.reg** to change the *instance_name* to the correct value for the Library path, save the file and exit.

Note: Because most Epiphany files are installed as read-only files, you need to change permissions before editing.

Step 2: Run **EpiPerfMon.reg**. To check the results, open the Registry and navigate to the key specified in Step 2. Ensure that the path supplied for **Library** is correct. If it is not, the Epiphany Channels object entry does not appear in the object list for the Performance Monitor.

Step 3: From a command line, go to the **Win32** directory of the current instance and run the following commands to unload parameters from any previous Registry entry and initialize the values associated with your Epiphany application:

```
unlodctr Epi
lodctr EpiPerfmon.ini
```

Note: Epi is the default driver and key specified in **EpiPerfmon.reg**.

You can now start the NT Performance Monitor and monitor the progress of data-extraction jobs performed by EpiChannel.

VERIFYING YOUR INSTALLATION

When you install the Epiphany software, the first screen displays your system configuration.

After installing the Epiphany software you can:

- Run the Epiphany tools, such as EpiManager, to configure your EpiCenter and set up EpiChannel extraction jobs. You will also use EpiCenter to construct the ticksheets that enable users to query the data in your data warehouse.

Migrating From Release 3.2 or 3.3

You must replace all references to these data types with references to one of the following new physical data types before you attempt to upgrade to the current release of Epiphany software:

EPIINT
FACTMONEY
FACTQTY

The steps for upgrading your EpiMeta metadata to the current release are as follows:

Step 1: Using Release 3.2 or 3.3 EpiCenter Manager, generate a full export file via the EpiCenter/Export/Export All command in the EpiCenter Manager menu.

By default, the Actuals and Aggs are not exported when you issue the Export All command. You must select both of these options in the runtime section of the Exporting Metadata dialog box to upgrade from your previous release to Release 3.4. (See Figure 3.) Selecting these options allows you to incorporate this essential metadata into the export file.

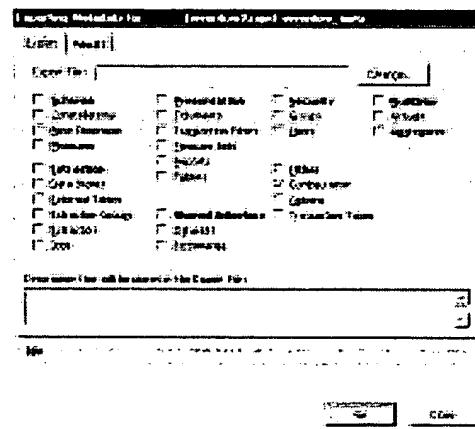


Figure 3: Exporting Metadata Dialog Box

MIGRATING FROM RELEASE 3.2 OR 3.3

This chapter describes the steps you must take to migrate metadata from Epiphany Release 3.2 or 3.3 to Release 3.4.

EpiMeta is the physical database that contains Epiphany control information. The EpiMeta database is implemented as a relational database with many tables and integrity constraints. All installations of a specific Epiphany release use the same data model for EpiMeta. However, upgrades to the product can include changes and additions to that model. Release 3.4 includes such changes. The remainder of this chapter describes the steps you must take to migrate your existing 3.2 or 3.3 EpiMeta data to the new data model in 3.4.

EpiManager provides a facility for upgrading your existing installation. Rather than operating on the EpiMeta database itself, the upgrade procedure operates on a standard Epiphany export file in Microsoft Access format.

Note: Beginning with Release 3.4, the following data types are no longer supported by Epiphany:

- NUMBER(9)
- NUMBER(9,2)
- NUMBER(9,5)
- FLOAT

Migrating From Release 3.2 or 3.3

Step 2: Build a new 3.4 EpiMeta database using the Initialize EpiCenter command in EpiManager, Release 3.4.

Step 3: Using the EpiManager, start importing the Release 3.2 or 3.3 export file. Specify a new file name for the “transformed” export file as prompted.

The import should proceed normally.

The transformation that occurs in Step 3 on page 51, modifies the previous export file such that all new metadata for Release 3.4 is included in the new Microsoft Access database. This modification is irreversible, which is why a new file name is requested. The export file is changed into a valid Release 3.4 export file, with the same format and information as would be contained in an export file produced using the Release 3.4 EpiCenter Manager export facility.

Note: The following caveats apply when you migrate from Release 3.2 only:

- The maximum length of names for fact and base dimension tables has been reduced from 25 to 20 characters. If these names in your EpiCenter Manager exceed 20 characters, reduce the number of characters in Release 3.2 before performing the Release 3.4 upgrade.
- The Saved Queries naming convention reflects the more rigid folder hierarchy in Release 3.4. (Folders must have unique names.) For this reason, when importing Release 3.2 saved queries into Release 3.4, a special folder called Release 3.2 Queries is created in the Public root folder. All Release 3.2 saved queries are placed in this folder; the names are altered slightly to ensure uniqueness. You can use the Report Gallery administration feature of EpiManager to move these folders into the appropriate Release 3.4 folder.
- Measure terms can no longer use MIN, MAX, and AVG in Release 3.4. Modify any Release 3.2 measure terms using these operators before upgrading.

Copyright and Trademarks

Copyright © 1998-1999 by Epiphany Marketing Software, Inc. All rights reserved.

This document and the software it describes are furnished under license and may be used or copied only in accordance with such license. Except as permitted by such license, the contents of this document may not be disclosed to third parties, copied, or duplicated in any form, in whole or in part, without the prior written permission of Epiphany Marketing Software, Inc.

This document contains propriety and confidential information of Epiphany Marketing Software, Inc. The contents of this document are for informational use only, and the contents are subject to change without notice. Epiphany Marketing Software, Inc. assumes no responsibility or liability for any errors or inaccuracies that may appear in this document.

Unpublished rights reserved under the copyright Laws of the United States.

Clarity™, Relevance™, Momentum™, and Magnitude™ are trademarks of Epiphany Marketing Software, Inc. All other products or name brands are trademarks of their respective holders.

Printed in the USA

Restricted Rights Legend

Software and accompanying materials acquired with United States Federal Government funds or intended for use within or for any United States federal agency are provided with "Restricted Rights" as defined in DFARS 252.227-7013(c)(1(ii)) or FAR 52.227-19.



E P I P H A N Y

SYSTEM GUIDE

3.4



Service Pack 1, March 1999

CHAPTER 2	39
EPIPHANY DATABASE EXTRACTION	
Extraction Phases: An Overview	41
Load Phase	41
Data Merging	42
Aggregate Building	42
Data Stores	43
The Role of Jobs in Extraction	45
Extractors	46
SQL Statement Extraction Step	48
SQL Macros	49
Staging Tables	49
External Tables	50
Semantic Instance Extraction Step	51
Applying a Semantic Type	51
System Calls	53
Aggregate Building	55
The Aggbuilder Program	58
The Aggregate Building Process	59
MomentumBuilder	60
Verifying MomentumBuilder Extraction	62
Custom Fact Indexing	62
The UNKNOWN Dimension Row	64
Referring to the UNKNOWN Row during Extraction	65
The CountUnjoined Semantic Type	65
Normal Extraction Order	66
Running Jobs: EpiChannel	67
The EpiChannel Command Line	67
EpiChannel Registry Keys	69
Output Files	70
Extracting New Rows Only	70
How EpiChannel Identifies Data To Be Extracted	72

CONTENTS

INTRODUCTION	13
About Epiphany	13
About Release 3.4	14
About This Guide	14
How To Use This Guide	16
Using Full-Text Search with PDF Files.	16
 CHAPTER 1	
BASIC CONCEPTS	19
Epiphany Database Schema	19
The EpiCenter	21
Dimension Tables	23
Dimension Roles	24
Date and Transtype Dimensions	24
Attributes	25
Fact Tables	25
Epiphany System Overview	27
The Epiphany System Is Metadata	29
The Uniform Treatment of Time	31
Uniform Transactional Data	32
Adaptive Architecture	35

Custom Fact Indexes	111
Fact Aggregate Browsing	113
Aggregation and Aggregate Grouping	115
The Default Aggregate Group	117
Measures	118
Defining a Measure	118
Measure Terms	120
Reverse Polish Notation	122
Ticksheet Types	124
Clarity	124
Relevance	124
Momentum	125
Configuring a Clarity or Relevance Ticksheet	126
General Tab	126
Assigning the Ticksheet to a Dataset	127
Assigning Access to Groups and Users	129
Attributes Tab	130
Setting Up a Glossary Entry	134
Filters Tab	134
Defining Filter Groups	137
Defining Filter Elements	138
Measure Mappings Tab	140
Adding Elements to Columns	141
Mapping Elements to a Measure	143
Verifying that All Elements are Mapped	143
Reports Tab	144
Copy Ticksheet Items	145
Configuring Relevance Ticksheets	146
Creating Default Relevance Ticksheets	147
Best & Worst	147
Profiling	147
Trends	148
Quarter Projections	149
Lifecycles	149

EpiChannel Debugging	74
Job Output	75
Error Messages	78
EpiChannel Debugging Levels	78
Setting Breakpoints	79
Trial Runs	79
EpiChannel Output	80
Log and Working Directories	80
Log Files versus Log Databases	81
Monitoring Jobs	81
Running the Performance Monitor	82
Mirroring: A and B Tables	83
SQL Limits for Facts	84

CHAPTER 3

EPICENTER MANAGER

85

Planning Your EpiCenters	86
Getting Started	87
Populating the Date Dimension	91
Working with an Existing EpiCenter	92
The EpiCenter Manager Window	93
Working with EpiCenter Manager	93
EpiCenter Manager Report	95
Setting Up an EpiCenter	97
Configuration	97
Base Dimension Tables	98
Defining Dimension Aggregates	102
Multiple Dimension Column Sets	105
Dimension Aggregate Browsing	105
Setting Up a Constellation	106
Defining Dimension Roles	107
Degenerate Dimensions	108
Defining Fact Tables	108
Defining Fact Columns	110

Setting Up Clusters and Counts on Fact Tables	173
Defining Transaction Filters	174
Default Settings on Momentum Ticksheets	176
Editing, Deleting, and Duplicating Ticksheets	176
Extraction	177
Data Stores	177
The Data Store Dialog Box	177
Modifying the Default Data Stores	180
Extractors	180
The Extractor Steps Dialog Box	182
Jobs	189
Adding Extractors and System Calls as Job Steps	192
External Tables	194
Momentum Extraction	195
Configuring E-Mail	196
Verifying that E-mail Notification Works	197
Configuring Outlook Exchange for EpiChannel E-mail Notification	
197	
Truncating Tables	198
Purging EpiMart Tables	199
Security	200
Setting Up Groups	201
Setting Up Users	205
Report Gallery	209
Folder Menu	211
Report Menu	211
Setting Permissions	212
Shared Interface	213
Setting Up Datasets	213
Generating Schema	214
Populating the Date Dimension Table	216
Exporting/Importing Metadata	217
Toggle A and B Tables	221
Web Builder	221

Aggregates for Relevance	150
Quarter Projections	150
Profiling	150
Influence	151
Best & Worst	151
Trends and Lifecycles	151
Relevance Influence Ticksheets	152
How Influence Works	152
Classification Trees	152
Regression Trees	153
Setting Up an Influence Ticksheet	153
Determining the Primary Dimension	154
Determining the Target	154
Determine the Source Attributes	154
Measure Sets	156
Setting Up Targets for the Ticksheet	158
Tips for Using Influence	159
Using Momentum Lists as Influence Attributes	159
Influence and Slowly Changing Dimensions	160
Influence and Aggregates	160
Performance Issue	160
Using Momentum	160
The Ind_Group_Joiner Fact Table	164
How to Populate the Ind_Group_Joiner Table	164
The Momentum Constellation	165
Why Use Adjacent Constellations?	166
Setting Up Momentum Adjacent Constellations	166
Additional Momentum Configuration	167
Mini-Dimensions	168
Transaction Filters	169
Clusters and Counts	170
Configuring a Momentum Ticksheet.	171
Setting Up Momentum Attributes	171
Filters	172

APPENDIX A	
EPIPHANY MACROS	257
System Call Macros.....	257
System Call Macro Syntax	258
Epiphany SQL Macros	262
Oracle-specific SQL Macros	262
Vendor-independent Macros	263
SQL Macro Usage	264
SQL Macro Notes	265
APPENDIX B	
EPICENTER CONFIGURATION	285
General Settings	285
Transaction Types	288
Measure Units	289
Option Labels	290
Momentum Labels.....	291
Ticksheet Types.....	293
APPENDIX C	
DATE DIMENSION FIELDS	295
APPENDIX D	
PHYSICAL TYPE VALUES	299
APPENDIX E	
WRITING STAGING SQL STATEMENTS	303
Base Dimension Staging SQL Statements	303
Duplicate sskey's	306
Dimension Staging Queries with Joins	306
Constructing Base Dimension Queries with DISTINCT Fact Values	307
Fact Staging SQL Statements	307
Using External Tables as Inputs to Staging Queries	310

Security Manager	222
Running the Scrutiny Debugging Tool	223
CHAPTER 4	
EPIPHANY APPLICATION SERVER	225
Starting and Stopping the Server	227
Running as a Service	227
Determining if the Application Server Is Running	228
Running as a Console Application	230
For Authentication to Work	231
Command-line Arguments	232
Refreshing the Application Server	232
The Refresh Command Line	235
Invoking RefreshApp	236
The EpiAppService Program	237
Command Syntax	238
EppiAppService Command Examples	240
The Application Server's Registry Keys	240
The Epiphany Proxy	243
Proxy Logging	243
Application Server Logging	245
Log File Location	246
Log File Naming Conventions	246
The Server Log	247
The Security Manager	248
Save and Restore Manager	249
Epiphany Applications	249
Application Server Security	250
Authentication Modules	252
Authentication Module Tips	253
Administrator Groups	256

VirtualMartDatabase Key Missing Error	334
Invalid Object DATE_O Error	334
Not a Valid Application Error	335
Internal Windows NT Error	335
EpiQuery Engine Database Connection Open Failure Exception ..	336
Charts Do Not Display	337
GIF Images Fail to Display on Web Pages	337
No Results Available for a Query	338
Result Page Error: Extraction Date Unknown	339
Web Server Message: Object Not Found	340
Browser Crashes When Retrieving Results from Application Server	341
Refresh Program Fails	342
Application Log Full Error	342
Application Server Log Security Problem	343

GLOSSARY	345
-----------------	------------

INDEX	361
--------------	------------

APPENDIX F	311
SEMANTIC TYPES	311
Dimension Semantic Types	311
Slowly Changing Dimensions	311
Latest Dimension Value	313
First Dimension Value	314
Initial Dimension Load	315
Fact Semantic Types	316
Count Unjoined (Optional)	316
Custom Fact Index	317
Transactional	317
Transactional/State-like	318
Transactional/State-like/Force Close	319
Pipelined	320
Initial Load Fact	321
Reload Max Date	322
APPENDIX G	323
EXPORT/IMPORT OF METADATA	323
Metadata Overview	323
Replacing Existing Metadata on Import	325
Actuals and Agg Metadata	326
Export File Format	326
APPENDIX H	329
TROUBLESHOOTING	329
Registry Editor Warning	329
SQL Server Error Message	330
Cannot Connect to the Server	330
Application Server Error Messages	330
User Cannot Log In	331
Additional Action to Take If User Still Cannot Log In	333
Service Control Manager Fails to Start Application Service	333
Out of Memory	334

About Release 3.4

Momentum List Manager gives you the ability to target customers quickly and easily and to generate lists of customers or individuals. Momentum's strength is that it can perform complex queries using data from multiple databases within your enterprise. A typical question that Momentum can easily answer is "Show me a list of all contacts at any customer who has purchased at least \$20,000 of products from our company in the past two quarters, has purchased at least one copy of Product A, has not purchased Product B, and is on a service contract." Construct your list by pointing and clicking in your Web browser.

ABOUT RELEASE 3.4

Epiphany Release 3.4 provides the following features and enhancements:

- Support for larger data sets than in previous versions
- Support for implementing an EpiCenter on the following database-server platforms:
 - Oracle8 on Solaris 2.6
 - SQL Server 7.0 on Windows NT Server 4.0

ABOUT THIS GUIDE

The *Epiphany System Guide* is intended for database administrators and consultants who set up and maintain an installation's datamart, as well as those who use Epiphany's Clarity, Relevance, and Momentum software to configure *ticketsheets* so that users can access relevant information in the datamart. (A *ticketsheet* is a form that allows end users to construct queries; it is called a *ticketsheet* because users select, or tick, items on a page.)

This *Guide* consists of the following chapters and appendices:

Chapter 1, *Basic Concepts*, describes the Epiphany database schema and introduces Epiphany-related data warehousing concepts.

Chapter 2, *Epiphany Database Extraction*, describes the Epiphany database extraction process.

INTRODUCTION

ABOUT EPIPHANY

Epiphany is the leading provider of Relationship Management (ERM) applications. Epiphany provides complete solutions for managing customer relationships by providing Web-based access to relevant information that is captured by various departments within a business organization. Epiphany applications allow you to browse through this information, gain new insights, and explore relationships and trends that might otherwise remain hidden within your various databases.

With packaged applications from Epiphany, you can:

- Browse current data from throughout your organization
- Drill down to analyze specific situations in detail
- Act on that information to satisfy current customers and cultivate new ones

The Clarity™, Relevance™, and Momentum™ suite of applications from Epiphany Marketing Software work in conjunction with your Epiphany data warehouse (the *EpiCenter*) to give you the ability to access, analyze, and act upon the data available within your enterprise.

Clarity gives you an instant, integrated view of your company's "information capital," which can include customers, products, leads, orders, and support calls. Whatever information exists in your enterprise can be brought into Epiphany and viewed using Clarity and a standard Web browser. Simply point your mouse, make selections, and click to access the data you want in report or chart format.

Relevance gives you rapid insight into your data, allowing typical end-users to put the power of technology to use in analyzing their data. Finding trends, projecting quarterly results, profiling and segmenting customers, and finding correlations and anomalies in your data is what Relevance is all about. Point and click in your Web browser to see this information in various graph formats.

HOW TO USE THIS GUIDE

A suggested approach to using this *Guide* follows:

- Step 1:** See the *Epiphany Installation Guide* for instructions on installing and configuring the software and setting up your system.
- Step 2:** Read Chapter 1, *Basic Concepts*, and Chapter 2, *Epiphany Database Extraction*, for the background material you need to set up an EpiCenter. Refer to the *Glossary* for definitions of terms.
- Step 3:** Follow the instructions in Chapter 3, *EpiCenter Manager*, to configure your EpiCenter and to construct ticksheets for your organization.
- Step 4:** Run the EpiChannel program, as explained in Chapter 2, *Epiphany Database Extraction*, to extract data from your source systems and place it into the Epiphany tables.
- Step 5:** Start the Application Server as described in Chapter 4, *Epiphany Application Server*, and verify that it is working.
- Step 6:** Refer to the appendices for more detailed information as directed throughout this *Guide*.

USING FULL-TEXT SEARCH WITH PDF FILES

The PDF file versions of the Epiphany documentation include a full-text index (**index.pdx**), which is a searchable database of all text in the Epiphany PDF documentation. A full-text search is much faster and more precise than using the standard Find command to search a document.

You can open and view a PDF file if you have Adobe Acrobat Reader installed on your system. To use this index, however, you need to have either Adobe Acrobat Exchange (the Reader plus enhancements), or Adobe Acrobat Reader with Search.

Chapter 3, *EpiCenter Manager*, explains how to set up your organization's data warehouse, which is called an EpiCenter. It also describes how to create and modify ticksheets. At the front end, users open a ticksheet in a Web browser and select options for the kind of data they want to display.

Chapter 4, *The Epiphany Application Server*, is an operator's guide to the component of the Epiphany ERM application suite that processes all user requests and returns query data in HTML format.

Appendix A, *Epiphany Macros*, defines the Epiphany-supplied system calls and SQL macros.

Appendix B, *EpiCenter Configuration*, describes the configuration data for a default EpiCenter.

Appendix C, *Date Dimension Fields*, defines the date dimension fields used by the Epiphany system.

Appendix D, *Physical Type Values*, defines the database type translations for the physical types used by the Epiphany system.

Appendix E, *Writing Staging SQL Statements*, explains how to write SQL statements for base dimension tables and fact tables. Instructions for using external tables as inputs to staging queries are also given.

Appendix F, *Semantic Types*, describes the dimension and fact semantic types.

Appendix G, *Export/Import of Metadata*, presents an overview of the Epiphany system's use of metadata as a basis for a discussion of how the Epiphany export/import metadata feature works.

Appendix H, *Troubleshooting*, describes the Epiphany error conditions and error messages and suggests corrective action.

The *Glossary* defines the terms used throughout this *Guide*.

Using Full-Text Search with PDF Files

Step 4: The PDF file is displayed with the first occurrence of your search text highlighted. Use the Search Next and Search Previous buttons in the toolbar to navigate to other occurrences. (For instruction on using Acrobat Exchange or Acrobat Reader, see their online guides, which are available from the Help menu.)

Acrobat Search has tools that enable you to expand and limit your search criteria. For complete instructions on using the search feature, see the *Search Online Guide*, also available from the Help menu.

Note: You can download the free Acrobat Reader with Search from Adobe's Web site (www.adobe.com/prodindex/acrobat/readstep.html). After you register with Adobe, select *Acrobat Reader with Search* from the pop-up menu at the same step of the process as you select the language and platform.

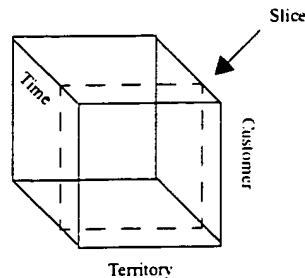
Follow these steps to tell Acrobat which full-text index to use:

- Step 1:** Open Adobe Acrobat Exchange or Acrobat Reader with Search.
- Step 2:** From the Tools main menu command, select Search\Indexes. The Index Selection dialog box displays available indexes. The index for both the *Epiphany System Guide 3.4* and the *Epiphany Installation Guide 3.4* is **index.pdx**. If it is not selected, you need to add it.
- Step 3:** Click Add to add the index for the Epiphany PDF files. Go to **C:\Program Files\Epiphany\instance_name\Docs** and double-click the file **index.pdx**.
- Step 4:** The index is added to the search list. Click OK to close the dialog box.

Follow these steps to search for a term:

- Step 1:** From the main menu, choose Tools\Search\Query.
- Step 2:** In the Find Results Containing Text box, enter the term whose occurrences you want to find in the document or documents. The term may be one or more words, or a number, and can include wildcards (an asterisk for multiple characters and a question mark for a single character).
- Step 3:** Click Search. The documents that contain text that matches your search query are listed in rank order in the Search Results window. Double-click the document that you want to search (usually the first one in the list).

Epiphany Database Schema



In the Epiphany system, an organization's data warehouse is known as an *EpiCenter*. (An organization may have multiple EpiCenters.) The Epiphany Application Suite consists of "front-end" Web-based applications, such as Clarity, Relevance, and Momentum, which are designed for database query and analysis. These applications allow users to query the EpiCenter by selecting dimensions and facts they want to know more about. The results of these queries (shown in reports, charts, lists, and graphs within the user's browser window) are derived from the intersection of these dimensions.

The EpiCenter represents a dimensional data warehouse with database tables organized in a *star schema* (see Figure 1, on page 21). At the center of a standard star schema is a *fact table* that contains measure data. Radiating outward from the fact table like the points of a star are multiple dimension tables. *Dimension tables* contain attribute data, such as the names of customers and territories. The fact table is connected, or joined, to each of the dimension tables, but the dimension tables are connected only to the fact table. This schema differs from that of many conventional relational databases where many tables are inter-joined.

An advantage of the Epiphany star schema is that it allows an organization's EpiCenter to be regenerated when its business model changes without the need to replace the data warehouse. The Epiphany star schema also facilitates the creation of new EpiCenters for an organization (data from existing EpiCenters can be imported).

CHAPTER 1

BASIC CONCEPTS

This chapter introduces the Epiphany database schema and provides an overview of the Epiphany system. Concepts integral to the Epiphany system, such as how it treats time and transactions, are also discussed.

EPIPHANY DATABASE SCHEMA

The Epiphany database schema is based on the dimensional data warehouse model. A data warehouse transforms the raw data from an organization's source system databases into data accessible for query and analysis. The data conforms to the organization's business model and is consistent, reusable, and flexible (that is, the data may be re-sorted by any measure the business uses). A data warehouse also provides the tools needed to query, analyze, and publish this data.¹

The dimensional data warehouse organizes data in what may be visualized as a cube. One can figuratively slice along any dimension of the cube of data to obtain information about the intersection of the dimensions at that point. For example, if the cube has the dimensions Customer, Territory, and Time, and one selects order amount as the measure, one could slice through the cube on the Time dimension to determine the total order amount by customer or territory at a specific date.

¹ *The Data Warehouse Toolkit* by Ralph Kimball (John Wiley & Sons, Inc., 1996) is an excellent book about data warehousing.

The EpiCenter

The EpiMart consists of fact, dimension, and staging tables. The fact and dimension tables contain actual customer data. Staging tables, which are discussed in Appendix 2, “Epiphany Database Extraction” are the first entry point of raw data from the source systems into the EpiMart—an interim stop before it reaches the EpiMart’s tables.

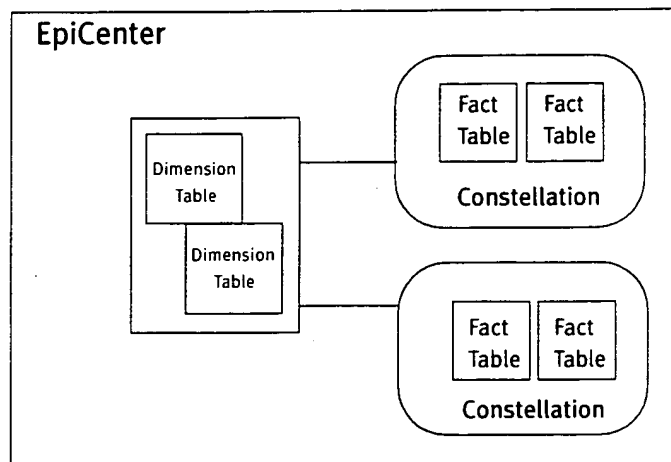


Figure 2: The Epiphany Star Schema

To facilitate the building of EpiCenters, Epiphany provides the EpiCenter Enterprise Manager, also called EpiCenter Manager. This is a Microsoft Windows application with an Explorer-like hierarchical structure (see Figure 3). The person who designs an EpiCenter (or EpiCenters) for a site uses the EpiCenter Manager’s graphical user interface to define the schema for the EpiMart tables.

The Epiphany star schema subsumes the standard star schema into a larger hierarchical organization (known as a *constellation*) that allows for the sharing of dimension tables by a set of similar fact tables. A constellation is a grouping mechanism for like-structured fact tables.

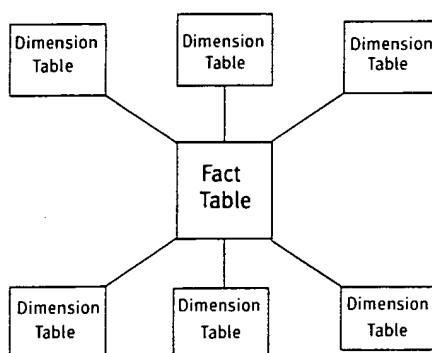


Figure 1: Standard Star Schema

As shown in the block diagram in Figure 2, on page 22, the EpiCenter is the top-level organizing principle. Each Epiphany site has at least one EpiCenter. An EpiCenter is organized into one or more constellations, and a constellation consists of a set of similar facts. Dimension tables are shared by multiple constellations within the EpiCenter.

The EpiCenter

An EpiCenter is composed of EpiMeta and EpiMart. EpiMeta refers to all of the Epiphany system's metadata tables. (*Metadata* is information about data, not data itself.) EpiMeta defines the schema for the EpiMart tables that will contain the actual extracted, organized data, such as customer, product, and order data. An EpiCenter is an EpiMeta database with its associated internal link to an EpiMart.

Dimension Tables

Dimension Roles

A row in a fact table may have several foreign keys that point to the same base dimension table, but the role usage differs. (A *foreign key* is a reference from one table to another table.) For example, a dimension role within an Order constellation and a dimension role within a Customer Support constellation may correspond to the same underlying Master Product List table for data. Within the constellation, these dimension roles may be assigned the role of Order Product List or Customer Support Product List, as appropriate. Multiple roles within a single constellation can refer to the same base dimension.

Dimension roles are defined within a constellation, and any fact tables in that constellation may inherit the dimension role. Dimension roles within constellations point to their associated base dimension table for their data. (A dimension role always has an associated base dimension table.)

Date and Transtype Dimensions

The Date dimension and Transtype (transaction type) dimension are assumed common to all constellations, and therefore are automatically provided as base dimension tables. To ensure consistent treatment of time, Epiphany uses a single date dimension throughout the system.

Transaction type dimensions are necessary because the EpiCenter must be able to distinguish among different rows in a single fact table, such as shipping transactions versus bookings, and bookings versus booked returns.

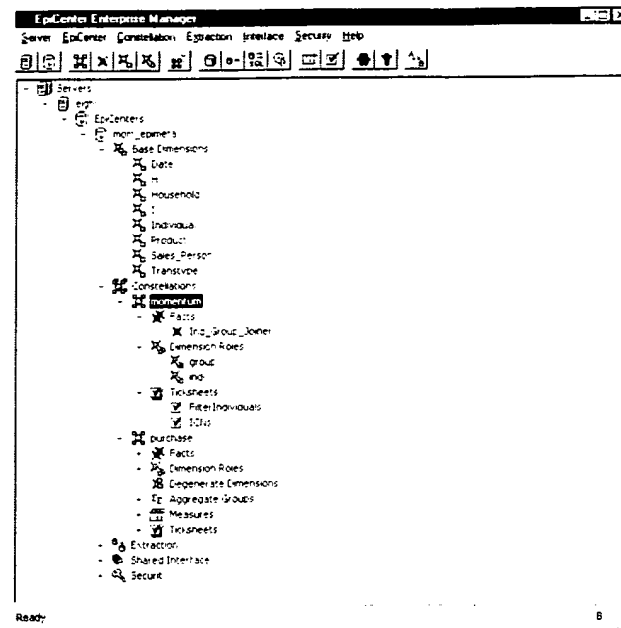


Figure 3: EpiCenter Enterprise Manager Window

Dimension Tables

A *base dimension table* consists of dimension columns that hold the actual attributes extracted from the source system. The Epiphany schema allows dimensions to be shared by all of the constellations within an EpiCenter. This is accomplished through the use of base dimension tables that serve as master tables for the entire EpiCenter. Each base dimension table may be referenced by multiple fact tables, or multiple times by the same fact table. These references are called *dimension roles*.

Fact Tables

Customer	Product	Territory	Date	Marketing Program	No. of Units	Unit_ Price	Unit_ Cost
Fact Foreign Key Fields					Fact Column Fields		

Figure 4: Sample Row in a Fact Table

For example, a fact table for Orders might have rows of data that contain fact columns and foreign keys that reference base dimensions, such as Business Unit and Territory. There may also be foreign keys for Customer_Bill-to and Customer_Ship-to that point to the Customer base dimension table. The dimension role metadata defines these foreign keys. (A fact table may have as many foreign keys to a single base dimension table as the business model requires.)

Each of the fact columns of a fact table can be totaled for any dimension and presented in report and chart format.

The Epiphany database organization allows users to make flexible queries. For example, a Clarity user can query the EpiMart for the names of customers who attended a Boston '97 trade show to determine the dollar amount of orders these attendees purchased as a result of this show.

Attributes

Attributes describe a dimension. Some attributes, such as Customer Name, are free-text descriptions, but most have a discrete set of values. For example, a Territory dimension table may consist of its territory key and text that describes each of the organization's sales regions; such as, Eastern, Western, and Southern. It may also contain attributes for a region's subdivisions, such as City and State. Therefore, a hierarchical relationship exists in the dimension table in which the City attribute rolls up (aggregates) into State, which then rolls up into Region.

Region	State	City
Western	CA	Los Angeles
Western	WA	Seattle

Note: In the EpiMart, facts are quantifiable measurements, usually numbers. Attributes of dimensions are usually character strings.

Fact Tables

Fact tables are physical database tables that contain dimension role foreign keys and fact columns. A fact column is a single column in the fact table whose values are numeric, such as *net_price*. In the EpiCenter, a measure is an arithmetic combination of fact columns.

A row in a fact table represents a relationship among a set of values (each value is a separate field in the table). As shown in Figure 4, a row may exist for a specific customer number, product number, territory number, and date—all foreign keys that point to base dimension tables for their data—and fact columns, such as the number of units and the price per unit. The table may have millions of rows of data.

Epiphany System Overview

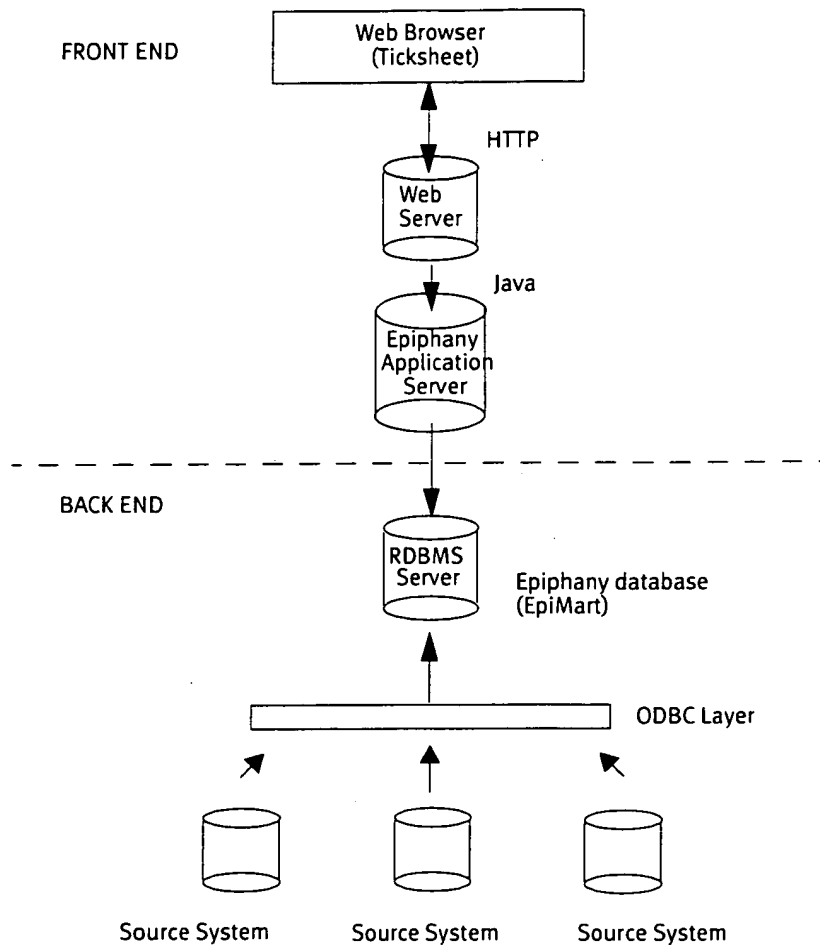


Figure 5: Epiphany System Overview

Although Epiphany's back-end tools can be used to generate and populate an industry-standard star schema, the power of the system derives from its integration with Epiphany's end-user Enterprise Relationship Management (ERM) applications. In order to achieve a robust, consistent application suite, each component interacts with a common metadata repository called EpiMeta.

EPIPHANY SYSTEM OVERVIEW

An Epiphany site's source data may reside in any number of database management source systems, both relational and non-relational. Examples of supported relational database management source systems (RDBMS) and their related software are—

- telemarketing applications (ACT! and Aurum)
- sales force automation (Siebel and Onyx)
- enterprise resource planning (Oracle Financials, PeopleSoft, SAP R/3)
- customer support applications (Clarify and Vantive)

As shown in Figure 5, at the back end of the Epiphany system, data flows from the source databases into the RDBMS server database where the Epiphany database, called the EpiMart, resides. The Epiphany system accesses the source data in a read-only fashion; no changes to an existing source system are necessary. Database updating occurs as part of the database extraction process, which is usually run on a nightly basis.

At the front end, users open a ticksheet in a Web browser on any computer system that supports a JavaScript-enabled browser and select options for the kind of data they want to display. (A ticksheet is a user-interface form in Epiphany that allows end users to construct queries; it is called a *ticksheet* because users select, or tick, items on a page.) The Web browser communicates with the Web server, which runs a Java program that queries the RDBMS server for the relevant information. The report is quickly generated and presented as an HTML document. The Epiphany Application Server is the Windows NT Service that connects with a Web server and delivers Epiphany ticksheets and reports. In the Epiphany system overview, it logically sits between the RDBMS and the Web server.

The Epiphany System Is Metadata

EpiMart contains the customer data at an implementation. The schema of tables in EpiMart is completely determined by the schema metadata in EpiMeta. The contents of EpiMart tables are determined by the instructions contained in the extraction metadata tables (and the relevant contents of the source system). Theoretically, the contents of EpiMeta can be used to reconstruct an equivalent EpiCenter; that is, re-extraction from the source system using the same EpiMeta should result in an identical EpiMart. Because EpiMeta effectively defines an EpiCenter, Epiphany provides an export/import utility to back up metadata, or to transfer subsets of metadata between EpiCenters. (see Appendix G, “Export/Import of Metadata” for more information.)

Each Epiphany tool reads from and/or writes to EpiMeta. The interaction between tools and EpiMeta is described below:

EpiCenter Manager Writes schema metadata, extraction metadata, security metadata, and aggregate definition metadata. EpiCenter Manager is also used for configuring user-interface metadata, including measures, ticksheets, and dictionary entries. It reads schema metadata and writes measure and ticksheet metadata.

EpiChannel Reads schema, extraction, semantic metadata.
Writes logging data about the extraction.

Aggbuilder Aggregation, the pre-calculation of selected computations involving facts to speed up the front-end query process, is performed by the Aggbuilder program. Aggbuilder reads schema metadata and aggregate definition metadata, and writes aggregate navigation metadata.

Aggregate navigation is the process by which the Epiphany query machinery determines the optimal aggregate table to use to satisfy an application’s request for information. (The *query machinery* is the component of the Epiphany Application Server that communicates

The Epiphany System Is Metadata

EpiMeta is built on top of a relational database engine. EpiMeta's tables store all persistent aspects of an Epiphany implementation. When you configure an Epiphany system (using EpiCenter Manager), these are the tables you write to. Each of Epiphany's ERM applications is a consumer of this metadata, which determines the appearance of the applications, as well as other aspects of their behavior.

THE EPIPHANY SYSTEM IS METADATA

All of the control information for an EpiCenter is stored in a single metadata repository called EpiMeta. EpiMeta represents a transactional, fully relational model of over one hundred and fifty tables that have complicated declarative referential integrity constraints. Epiphany provides the EpiCenter Manager tool for configuring this metadata without the need to write to, or even know about, the underlying data structures.

The EpiMeta tables of metadata control all aspects of the system including:

- Star schema structure
- Extraction workflow and semantics
- Aggregation and other performance enhancements
- Business calculations (or measures)
- User interface layout
- Security information
- Saved Report objects

Uniform Transactional Data

The queries that are constructed for these purposes do not perform date arithmetic, such as:

```
SELECT * FROM MyFact
WHERE date_key > '6/1/1998' and date_key < '7/1/1998'
```

Queries of the above form become unwieldy because of the non-uniformity of the calendar with respect to the number of days in the month, leap years, weekly overlaps with months, and so on. Instead, determining which days belong to which weeks, months, quarters, or years is done once when the date dimension is populated. This table is then fully enumerated with all values of interest for the EpiCenter, allowing queries of the form:

```
SELECT * FROM MyFact, Date_0 WHERE MyFact.date_key =Date_0.date_key
AND cq_and_cy_name = 'Jun 1998'
```

Queries of this form can easily take advantage of RDBMS indexes.

Epiphany allows for specification of Calendar fiscal quarters, as well as 13-week manufacturing calendars (4-4-5 calendars), in which a quarter always starts on the same day of the week. The date dimension can also be configured to specify which weekdays start and end a business week (for example, Sunday through Saturday versus Monday through Sunday).

UNIFORM TRANSACTIONAL DATA

A transaction represents an event in time. All data in an EpiMart fact table is stored in transactional format. For certain business entities, this format makes intuitive sense, such as in the case of an invoice in which a record in EpiMart represents the shipment of a product to a customer at a certain point in time. For the most part, this event cannot be changed; it simply happened and is stored as such in an EpiMart fact table.

Other business entities are not so easily made into transactions. When a customer calls to order a product, he might choose to order 10 units. In the Epiphany system, this fact is entered as a transaction with quantity 10 for Product P1 to Customer C1.

with EpiMart and actually issues SQL statements against the DBMS.)

Epiphany Application Suite

Clarity, Relevance, and Momentum applications read schema, measure, ticksheet, and aggregate navigation metadata. They write saved object metadata.

Each table in EpiMeta defines an integer primary key. The database engine provides the actual unique primary key values for each row in the metadata. All foreign keys between metadata tables are accomplished with these integer columns. The benefit of using non-natural primary keys is that all other columns in the metadata can be changed without affecting relationships in the model. Note that each foreign key uses declarative referential integrity to ensure consistent metadata. Additionally, EpiMeta enforces other declarative integrity constraints based on the correct interpretation of these tables.

THE UNIFORM TREATMENT OF TIME

In many Enterprise Relationship Management (ERM) applications, the time on which a fact occurs is a crucial dimension. It can be complicated, however, to create the semantics of calendars and other date attributes for a new data warehouse. Epiphany provides a time dimension via the special table *Date_0*, along with EpiCenter Manager, for populating the date dimension with the parameters that make sense for each implementation. Each row in this table represents a single day and the attributes associated with that day.

Every fact table in EpiMart automatically contains a foreign key to this special date dimension. Many of the fact semantic templates (the generic programs that perform business transformations on extracted data) depend on the presence of this field. By joining the fact table to the date dimension, Epiphany's ERM applications are able to answer questions of the form:

- Which facts occurred this quarter?
- What is my weekly backlog for the year?
- What products were bought last month?

Uniform Transactional Data

In Epiphany's fact table the same information is represented transactionally as:

Inventory Fact

Customer	Product	Date	Change In Inventory
C1	P1	6/1/1998	10
C1	P1	6/8/1998	2
C1	P1	6/15/1998	-7

Of course, an end user who asks for a report of inventory by week wants to see the data reported in Clarity as it is reported by the source system. To accomplish this, use EpiCenter Manager to define a measure that has a backlog type that makes this measure an accumulator over time. This causes the transactions to be reassembled back into the reported format.

Why go to the trouble of disassembling orders and inventories into transactions, only to reassemble the previous format at output time? The reason is that transactions are the most additive way to store data, which means that transactions can be recombined along arbitrary dimensional boundaries.

The way in which Inventory data is reported above can easily answer queries of the form "Inventory by *week*." This data, however, cannot easily be used to answer queries by month or year without some external knowledge of which week is the end of the month. Today's RDBMS engines cannot efficiently handle these types of queries without Epiphany's transactional storage. Note that the inventory for the month of June can be calculated by adding up all the transactions that occur between 6/1 and 6/30, yielding an ending inventory of $10 + 2 - 7 = 5$.

Important: All source system data must be made into transactions. Keep this in mind when you author an Epiphany extractor or choose a semantic type.

Uniform Transactional Data

Order Fact

Customer	Product	Date	Quantity
C1	P1	6/1/1998	10

However, if the customer calls back the next day and changes the order to 15 units, then instead of entering a separate transaction of 15, Epiphany's extraction machinery enters the *difference* of 5 as a new transaction on that second day.

Customer	Product	Date	Quantity
C1	P1	6/1/1998	10
C1	P1	6/2/1998	5

Similarly, in the case of Inventory, Epiphany tracks changes in inventory as transactions instead of restating the reported inventory from the outside world. For example, if Customer C1 *reports* inventory for Product P1 as:

Reported Inventory in Source System

Date	Quantity On Hand
6/1/1998	10
6/8/1998	12
6/15/1998	5

Adaptive Architecture

The name of the table is taken from the name of the base dimension. Each dimension column becomes a physical column in the EpiMart table with the same name.

- Fact tables

The name of the table is taken from the name of the fact table. Each metadata fact column becomes a numeric column in the EpiMart table with the same name. Each dimension role and degenerate dimension of the constellation to which the fact table belongs become a foreign key column in the fact table. [A *degenerate dimension* is a single column dimension in a fact table that stores textual information in lieu of using a foreign key to a base dimension table. For more information, see “Degenerate Dimensions” on page 108.]

- External tables

The name of the table is taken from the name of the external table. Each external column becomes a column in the EpiMart table with the same name.

Each time Generate Schema is executed, EpiCenter Manager records the schema of the newly built tables in a set of metadata tables called the Actual tables. When subsequent changes are made to the schema definition, EpiCenter Manager examines these Actual tables to compute a delta operation.

Important: If Generate Schema is not run and the schema definition has been altered, neither Aggbuilder nor the Epiphany Application Server will start. Both will recognize a difference between the metadata schema definition and the contents of the Actual tables.

ADAPTIVE ARCHITECTURE

In traditional client/server application environments, changes to an underlying table's schema adversely affect programs that operate on that schema. The goal of Epiphany's Adaptive Architecture is to allow on-the-fly changes to the EpiMart schema while preserving the form and proper operation of the entire Epiphany Application Suite.

A schema change in EpiMart can affect these components:

- Extraction statements that populate the staging tables.
- Semantic instance programs that merge staging data with EpiMart data.
- Aggregates defined on these tables.
- Measures that refer to these tables.
- Ticksheets that refer to columns in these tables.
- Security restrictions on columns in these tables.

The use of a single metadata repository (EpiMeta) ensures that all components of the system receive notice of a change simultaneously; for example, Aggbuilder will not try to build aggregates on tables or columns that no longer exist.

During extraction, semantic instances are SQL programs that accomplish business transformations on extracted data. Since these programs must be changed in response to schema changes, these programs are compiled on-the-fly at execution. Thus the program uses the latest schema metadata in its construction, which results in a well-formed set of SQL statements.

EpiCenter's Generate Schema command creates and modifies the EpiMart tables. The first time this operation is executed, all tables are built using CREATE TABLE statements. These tables include base dimension, fact, and external tables. Certain metadata fields are used in the actual construction of these tables.

EpiCenter Manager ensures that these fields follow the proper naming conventions for table and column names. These naming conventions are as follows:

- Base dimensions

Adaptive Architecture

Table 1: Schema Operations

Rename a dimension role	Treated the same as the deletion of the old column and an addition of the new one.
Add a new degenerate dimension	The column is added to all fact tables in that constellation. All existing rows get the special value UNKNOWN.
Delete a degenerate dimension	The column is removed from all fact tables in that constellation.
Rename a degenerate dimension	Treated the same as a deletion of the old column and an addition of the new one.
Add a new external table	The table is created from scratch.
Any change to an external table including changes to column	The table is dropped and recreated.

The following table describes the schema operations.

Table 1: Schema Operations

Operation	Action
Add a new base dimension	The table is created from scratch.
Delete a base dimension table	The table is dropped, and the dimension role columns of any fact tables that point to that base dimension table are deleted.
Rename a base dimension table	The table is renamed.
Add a new dimension column to an existing base dimension	The existing table is altered to include that column. All existing rows take the default value for that column.
Delete a dimension column from a base dimension	The column is removed.
Rename a base dimension column	Treated the same as a deletion of the old column and an addition of a new one.
Add a new fact table	The table is created from scratch.
Delete a fact table	The fact table is dropped.
Rename a fact table	The fact table is renamed.
Add a new fact column	The table is modified to have the new column. All existing rows get the value 0.
Delete a fact column	The column is removed.
Rename a fact column	Treated the same as a deletion of the old column and an addition of a new one.
Add a dimension role	The column is added to all fact tables in that constellation. The value for all existing rows is set to 1, which is a special value that always points to the UNKNOWN row in the base dimension to which that role refers.
Delete a dimension role	The column is removed from all fact tables in that constellation.

Epiphany Database Extraction

- **Semantics** (types/templates and instances). At a high level, a semantic type reflects the semantics of a company's business processes. During data extraction, semantic instances, which are post-compilation SQL programs, are used to merge data with the proper semantics into data loaded during previous extractions. See "Data Merging" on page 42 and "Semantic Instance Extraction Step" on page 51 for definitions of semantic-related terms.
- **EpiChannel**. This is the Epiphany extraction program that executes jobs and logs job activity. See "Running Jobs: EpiChannel" on page 67 and "EpiChannel Debugging" on page 74 for more information.
- **Database administration tools**. These tools are supplied by the database vendor to administer the physical characteristics of the databases and the interconnections among them. (Please refer to your database vendor's documentation for information about these tools.)

The major topics covered in this chapter are—

- *"Extraction Phases: An Overview" on page 41*
- "Data Stores" on page 43
- "The Role of Jobs in Extraction" on page 45
- "Extractors" on page 46
- "System Calls" on page 53
- "Aggregate Building" on page 55
- "MomentumBuilder" on page 60
- "MomentumBuilder" on page 60
- "Custom Fact Indexing" on page 62
- "The UNKNOWN Dimension Row" on page 64
- "Running Jobs: EpiChannel" on page 67
- "EpiChannel Debugging" on page 74
- "EpiChannel Output" on page 80
- "Monitoring Jobs" on page 81
- "Mirroring: A and B Tables" on page 83
- "SQL Limits for Facts" on page 84

EPIPHANY DATABASE EXTRACTION

The goal of the extraction process is to extract raw data from data source systems and to place it into EpiMart tables where the data is accessible for query by front-end users. The extraction process, which is usually performed on a nightly basis, involves the use of these components:

- **EpiCenter Manager.** In addition to using EpiCenter Manager to define your site's star schema and constellations, you will use it to define extraction jobs. See "Extraction" on page 177 for instructions.
- **EpiCenters (EpiMeta and EpiMart).** EpiMeta is the metadata database that defines the star schema and the semantics applicable to an organization. (*Semantics* refers to the means by which data is interpreted for an organization in terms of its business processes.) EpiMeta also contains data related to the extraction jobs.

EpiMart consists of fact, dimension, and staging tables. The fact and base dimension tables hold the data of the star schema (actual customer data), and the staging tables help to populate them.

Data Merging

Data Merging

The second main phase of extraction is data merging. After the staging tables are fully loaded, semantic instances merge the new data in the staging tables into the EpiMart database tables. A *semantic instance* is the usage of a generic semantic type on one fact or dimension table during part of an extraction job. As part of defining tables, you assign each an Epiphany-defined semantic type. A *semantic type* refers to the logical business process for which a generic SQL program called a *semantic template* will be applied.

The Epiphany extraction program, EpiChannel, executes the semantic instance at the appropriate time during an extraction job.

Aggregate Building

The third and final extraction phase is aggregate building. Aggregates are pre-calculated and pre-stored summaries of data that significantly accelerate the front-end *query machinery*. The query machinery is the component of the Epiphany Application Server that communicates with EpiMart and issues SQL statements against the RDBMS.

You will use EpiCenter Manager to define which facts are aggregated for groups of dimensions. The aggregate builder program (**agg.exe**), which typically runs immediately following the data extraction, performs roll-ups, or aggregations, on the groups and includes them as aggregates in the EpiMart tables.

At the conclusion of the extraction phase, the extracted data resides in the proper tables and columns in the EpiMart. The data has been extracted from the source system or file data source (called a data store) and placed in the data store on the RDBMS server.

EXTRACTION PHASES: AN OVERVIEW

There are three main phases of the extraction process—load (pull/push), data merging, and aggregate building. Each phase is related to the metadata that defines the tables and their columns.

Load Phase

During the load, or pull/push extraction phase, data is extracted from a source database or file in raw format and loaded into interim tables (staging tables or external tables). A *staging table* is a table that contains all of the fields required by a single base dimension or fact table. Staging tables hold the data in preparation for the second phase of extraction: data merging.

An *external table* is a table built in EpiMart (and thus internal to the Epiphany system), which usually serves as a temporary table during extraction. The external table, which is similar to a staging table, receives the data directly. External tables serve as intermediary tables when more complicated multi-staged extraction is required.

Staging and external tables have a schema similar to the target EpiMart tables (Epiphany's Adaptive Schema Generator generates both simultaneously). The *Adaptive Schema Generator* is a component of EpiCenter Manager that builds the tables in EpiMart using the schema metadata definitions in EpiMeta.

When you run the EpiChannel program to start the extraction process, it reads the metadata for the job you selected and begins executing extractor steps. An *extractor* logically specifies a series of steps that move data from the input to the output data source. Occasionally, an extractor step is directed to populate external tables rather than staging tables.

Data Stores

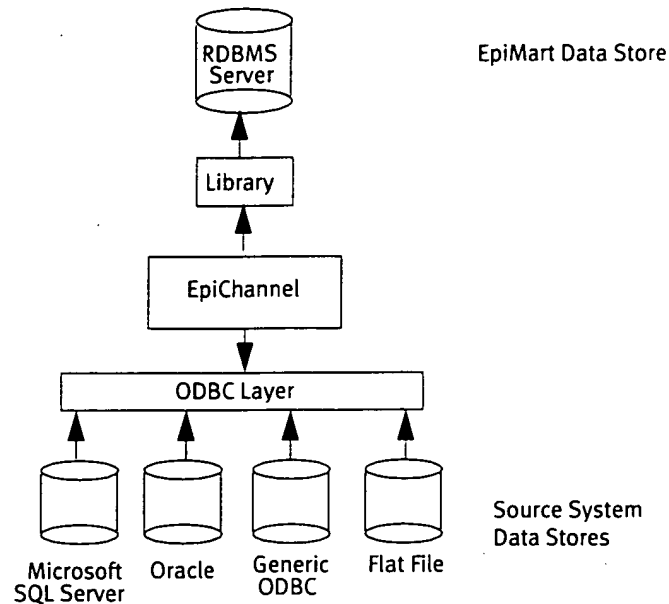


Figure 6: The Back End of the EpiPhany System

- *LoggingDB*

Specifies the data store for the EpiChannel logs.

You may, however, set up as many data stores in EpiCenter Manager as necessary. For example, operational systems for different divisions within an EpiCenter may have the same fact and dimension tables (EpiMart), but draw data from different source systems. In this case, you would create a data store for each input source system. For more information, see “Data Stores” on page 177.

Note: The EpiMart is the data store, or database, that you set up for your data warehouse. The EpiMeta holds the description of your data (the extraction jobs, the ticksheets, and the star schema definitions). The EpiCenter is a combination of EpiMart and EpiMeta.

DATA STORES

A *data store* is a logical location of data that functions either as a source or sink within an EpiCenter. Data stores include relational database connections, as well as files and directories. The concept of a data store is integral to the Epiphany extraction process. A data store represents the physical location of a source of data that the Epiphany system can read from and/or write to. Typically, the input data store is a source system database or file, and the output data store is EpiMart, which holds customer data, on an RDBMS server. As part of the installation procedure, you will create a new, empty database for EpiMart (and EpiMeta).

As shown in Figure 6, the Epiphany system extracts the raw data from a site's source system—for example, a RDBMS server, a generic ODBC source system, or a flat file—via Open Database Connectivity (ODBC). ODBC is an abstraction layer that provides a common interface to many databases. This data is read by various ODBC drivers and written using the target database system's API. The Epiphany system can access data if an ODBC driver exists.

EpiChannel, the Epiphany extraction program that executes jobs and logs job activity, opens an ODBC connection to the source system and a database library connection to the target server and initializes tables.

You will use EpiCenter Manager to configure the data stores as part of setting up your site's extraction process. When an EpiCenter is created, it has three default data stores:

- *EpiMart*
A special data store for the actual database on which Epiphany's front-end applications will run.
- *JobFileLog*
Specifies the data store for EpiChannel job logs.

Extractors

EXTRACTORS

An extractor is one or more sets of SQL operations that will be executed against a particular source and destination data store. These same sets or groups of SQL operations may be shared by another extractor that applies them to different data stores. The SQL operations may be either SQL statements or semantic instances. In general, SQL statements are used to extract data, and semantic instances are used to merge data with the proper semantics into data that has been loaded during previous extractions.

Semantic instances are designed to work on an idealized star schema. Macros in these semantic instances are translated at extraction time to the tables and columns in the target star schema. (The fact or dimension table listed in the Semantic Instance dialog box (Figure 53, on page 188) in EpiCenter Manager determines which column names are actually used in the final SQL.) This enables the complex transformation logic in a semantic instance to be applied to star schemas other than the one for which the semantic instance was first used.

As shown in the job work flow (see Figure 8), a job step may be either a system call or an extractor. An *extractor* is a list of extractor steps (and input and output data stores) that move data from the input to the output data store. An *extractor step* is a single step of an extractor, which always points to an extraction group. An *extraction group* is a set of extraction steps. An *extraction step* is a single atomic extraction operation that can be either a SQL statement or a semantic instance.

Extraction steps and groups are both modular. After you define them, you may reuse them in other jobs. (The Extractor Steps dialog box (Figure 51, on page 183) in EpiCenter Manager is where you define a new, or select an existing extractor for a job.)

Note: An extractor has associated input and output data stores and exists independently of any extraction step.

THE ROLE OF JOBS IN EXTRACTION

The Epiphany extraction process works through the running of jobs. A site defines the jobs needed to extract its data and runs these jobs on a regular basis to update the EpiMart. A *job* is basically a batch of work to be performed as a unit. The work consists of an ordered list of job steps, each of which is either an extractor or a system call. System call job steps may be interspersed with extractor job steps in any order. (See Figure 7, on page 45 and Figure 8, on page 47.)

A simple extraction may consist of one job, which has multiple steps. Each step utilizes the data produced by the previous step. A multi-stage extraction, however, may consist of multiple jobs. The EpiCenter Manager provides a graphical user interface (GUI) for defining job steps and the job input and output data stores. The actual job steps (extractors and system calls) need to be customized for each site.

After the jobs have been defined and the EpiCenter schema generated, the database administrator can invoke the EpiChannel (**extract.exe**, or **extract**) command to begin the extraction process that will populate a new EpiMart, or update an existing one.

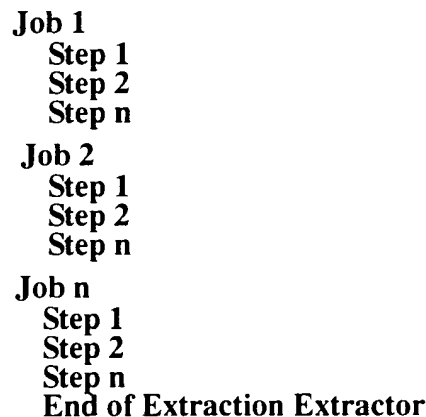


Figure 7: Job Work Flow

SQL Statement Extraction Step

SQL Statement Extraction Step

An SQL statement extraction step may be either a stand-alone SQL statement or a pull/push statement.

A stand-alone SQL statement is executed against either the source or destination data store for that extractor and is usually specific to the data store's type. These SQL statements are usually issued to achieve a side effect, such as dumping the transaction log or setting up environment variables. Any returned results are discarded.

A pull/push SQL statement is a type of extraction step that issues SQL against an input data store (and is interpreted according to the rules of that database's engine). The returned results are inserted (pushed) into the destination table for that step, either a staging table or an external table (the output data store of the extractor).

The result set columns are mapped either to columns in the destination tables or to other functions. The mapping is a case-insensitive exact match by name of the available columns, or function results, to the destination table columns.²

At this stage of the extraction, each destination table column must have a matched value. If not, the mapping fails, the SQL statement results are discarded, and the statement fails. Extra columns that are available from the SQL statement or functions, but which are not used in the destination system, are not considered to be errors. (SQL statements with extra columns will have their mapping displayed in the EpiChannel log if you set the verbosity level for EpiChannel high enough; see "EpiChannel Debugging Levels" on page 78.)

Once mapped, the rows from extraction SQL statements are fetched into EpiChannel memory where they are verified for field width and forwarded to any function that consumes the field. The mapped results are forwarded to the destination database.

² Special characters such as the at symbol (@) are removed when this match is performed.

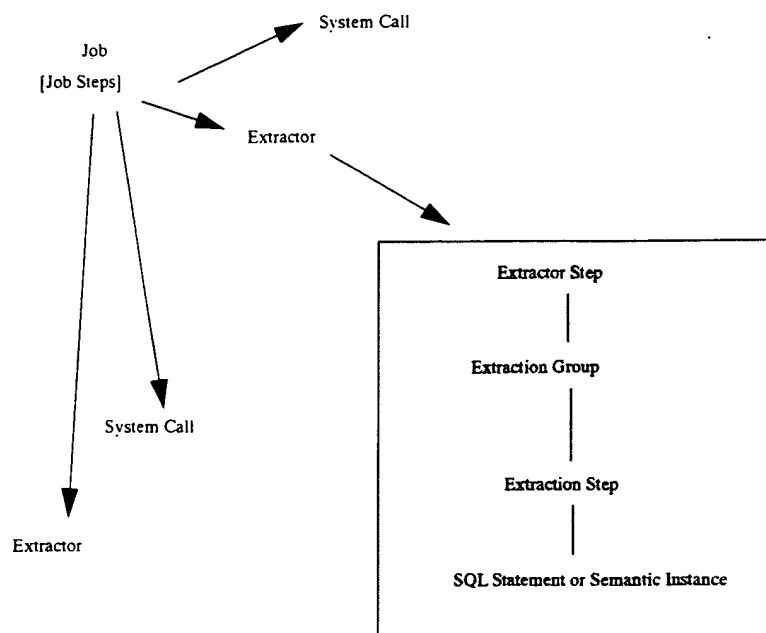


Figure 8: Job Structure

An extractor can move data from a single source data store to a single destination data store. A job may consist of multiple extractors, however, each with its own independent input and output data stores.

Jobs may also share extractors, applying the same logic in different orders—if the jobs' data stores have common structures. For example, a job that runs Monday through Friday could share the same extractor with a Weekend version of the same job in which the only difference is that the Weekend job has an extra step, an extractor that updates rarely changing tables.

The two kinds of extraction steps that may comprise an extractor step are discussed below: SQL statement and semantic instance.

All operations on the source data store or the EpiMeta data store occur through the appropriate native library (API) for the EpiCenter. The inserts are batched so that the extraction statements maintain a count of rows fetched, rows forwarded, and rows committed. The batch is fully sent and committed before the extraction SQL statement ends and the next job step may begin.

Any error, as determined by the database engine evaluating the SQL, results in the SQL statement being considered in error. If the action plan for the SQL statement (as selected in the General tab of the SQL statement dialog box) is to abort, then the job halts without executing any other SQL operations or job steps.

SQL Macros

A SQL statement may contain extraction macros that are expanded before the SQL is executed. Some of these macros provide database-independent functionality by expanding in different ways for different databases, and some use the metadata to cause the SQL to reflect the edits made in EpiCenter Manager. These macros are described in Appendix A, “Epiphany Macros.”

Staging Tables

The most common destination for an extraction statement is a staging table. As mentioned, a staging table contains all of the fields required by a single base dimension or fact table. Unlike the real fact and base dimension tables in the EpiMart, the staging tables have dimension columns that contain key fields from the customer’s system, rather than Epiphany-generated foreign keys. Staging tables typically contain data extracted by only one job, rather than accumulating data over time.

Once a staging table is populated, semantic instances are used to merge the new data in the staging table into the existing data. A semantic instance is a series of statements, similar to a program. These statements maintain the integrity and additive nature of the columns in the facts tables and maintain correct references to dimensions even though the dimension may change from one reporting of a fact to another.

See Appendix E, “Writing Staging SQL Statements” for more information.

SQL Statement Extraction Step

External Tables

Extraction statements are sometimes directed to load external tables rather than staging tables. External tables serve as intermediary tables when more complicated multi-staged extraction is required, such as for the grouping of data, dividing values in a field into bins, or joining with data from another source system. They may also serve as holding fields used by non-Epiphany data transformation tools, such as third-party name and address cleansing tools. In the case of cleansing tools, extractors merge the cleansed names and addresses with other tables extracted from the source databases into the EpiMart tables.

EpiCenter Manager provides the extractor named *End of Extraction* for each EpiCenter by default. This extractor consists of two steps:

Step 1: Issue SQL against the EpiMart to determine the date of the last extraction to be displayed to users.

Step 2: Toggle the current EpiMart from A to B, or vice versa.

Step 1 usually consists of SQL statements that load the external table named *last_extract_date*. The *last_extract_date* table is provided by Epiphany, and is recommended for use at your site.

Step 2 updates the *config_master* value *last_extract_date* with the latest value received from Step 1. It then updates the *config_master* value called *current_datamart*, switching it between A and B. (*config_master* is a metadata table that contains various system parameters in name/value format; these are modified via the Configuration dialog box in EpiCenter Manager.)

Note: The last extraction date setting is the date that appears as the *Data is valid as of...* in Clarity reports. It is determined by an option in the EpiCenter Manager's Configuration dialog box (described in Appendix B, "EpiCenter Configuration").

Switching the focus between the A and B tables involves an update to a single row in the database (no tables or databases are actually moved). See "Mirroring: A and B Tables" on page 83 for a discussion of A and B tables.

See "Using External Tables as Inputs to Staging Queries" on page 310 for more information about external tables.

Semantic Instance Extraction Step

As mentioned, the Epiphany extraction process (exclusive of aggregation) is a two-phase operation:

- load (pull/push) phase
The load phase refers to the loading of staging tables, which are the first entry points of data into EpiMart.
- data merging phase
The data merging phase refers to the loading of staging tables into the EpiMart's permanent base tables.

This process of moving data from staging tables into the EpiMart's base tables is called *semantic transformation*. Proper configuration of the data merging phase of extraction requires an understanding of the underlying meaning, or semantics, of the data.

Epiphany provides a set of semantic types/semantic templates that you may use without having to be concerned with their internal code. A *semantic template* is a generic SQL program intended to accomplish specific business rules during extraction (these rules are referred to as *semantic types*). A semantic template program contains SQL with generic table names; it does not refer to actual column or table names. Only when the semantic template is applied to a base dimension table or fact table does the SQL contained within it refer to actual column and table names.

When a semantic template is combined with a base dimension table or a fact table, the result is a semantic instance. A *semantic instance* is a valid SQL program that can be run against EpiMart.

Applying a Semantic Type

While SQL Statement extraction steps are used to move data from the source systems (non-Epiphany data stores) into temporary staging tables in the Epiphany datamart, semantic instance extraction steps move data from staging tables into fact tables and base dimension tables, where the data can be queried by the end user.

Applying a Semantic Type

To define a semantic instance extraction step, you will use the Semantic Instance dialog box in EpiCenter Manager to assign a semantic type to either a base dimension table or a fact table. Semantic types are rules provided by Epiphany that use business logic to regulate how new data from a staging table is merged with existing data in the base dimension table or the fact table.

The simplest semantic types (Initial Load Dimension and Initial Load Fact) simply overwrite the base dimension table or the fact table with the contents of the staging table; other semantic types deal intelligently with backlogs, changes in dimensions over time, and so forth when combining new and existing data. The different semantic types are described in Appendix F, “Semantic Types.”

You must use at least one semantic type for each base dimension table and fact table in your EpiCenter during extraction. Otherwise, new values for those base dimensions or facts that lack semantic types will not make their way into the EpiMart. You do not usually want to use more than one semantic type for a single base dimension or fact table since the semantic types generally cancel one another out. The exception is that certain semantic types for facts (Count Unjoined and Custom Fact Index) do not actually merge data from the staging table into the EpiMart. These semantic types have no effect on the base dimension table or fact tables and are instead useful for their side effects. These special semantic types should be used in conjunction with other semantic types that perform the actual merging of data.

Dimension semantic types have two purposes: creating base dimension tables and creating dimension mapping tables. Dimension mapping tables are used to convert dimension source system identifier keys (*sskey*) in fact tables to Epiphany dimension keys. Dimension semantic types are also responsible for proper indexing of these two tables.

Fact semantic types are normally used to populate and Index new fact base tables. See Appendix F, “Semantic Types” for a description of source system keys.

SYSTEM CALLS

While SQL-based transformations may be sufficient to extract from simple databases, sites with more complex databases may require multi-stages and additional commands, such as lookup tables, aggregation splits, gathering data into ranges (binning), and duplicate detection. For this purpose, you may use system calls, which are executed during a job as if invoked from the console's DOS command line.

In a multi-stage extraction, files might need to be moved, decompressed, de-encrypted, or purged. Although these steps could be placed before or after extraction statements, they may need to be placed in-between SQL-based steps. EpiChannel coordinates the execution of system calls between extractors.

A complex job might require the following steps:

- Step 1:** The invocation of a system call to uncompress a file.
- Step 2:** The invocation of an extractor to populate external tables.
- Step 3:** A system call to invoke third-party software to cleanse the names in the external tables.
- Step 4:** Extractors to merge the cleansed names (along with data extracted from other tables in the original database and other databases) into the EpiMart.

Many system calls need to reference the location of databases or files. In the above examples, Steps 1, 2, and 3 need to share the file and database names for the data they pass to each other. Although these names could be hard-coded, this is not the best approach; for example:

- The system call could break if the database login information changed, and the system call text was not modified accordingly.
- The system call cannot be easily reused.
- Some file names need to be adjusted from run to run so that useful intermediate data is not overwritten.

System Calls

Job data store roles provide an alternative to hard coding database and file information in system calls. Roles are referenced by macros in systems calls, and these macros look to the job to see what data stores are associated with the roles. The information from the matching data store is replaced by the macro. For example, the system call

```
mkdir $$DIRNAME[Working Dir]/aggbuilder_temp
```

would have the `$$DIRNAME` macro expanded to the MS-DOS file path that EpiChannel is using as a *working directory*. The **mkdir** command then creates a subdirectory in this location. Because EpiChannel generates a unique directory name for itself for each run, the above command would create a corresponding unique subdirectory for use by the Aggregate builder program (Aggbuilder), and this directory would be moved/purged along with EpiChannel logs.

The system call is executed from the working directory in the same way as if you opened a *command.com* window and changed directories to that location. You should either use an absolute path to specify the command, or you should make sure the value of the `PATH` environment variable includes the directory that contains the command executable.

`$$EPIBIN` expands to the path of the **extract.exe** directory, so if you enter:

```
"$EPIBIN/mycommand.exe" arg1 arg2 ...
```

and place **mycommand.exe** in the same **win32** directory as **extract.exe**, then it will work. If you do anything else, you are subject to the rules of DOS.

Note: Use quotes as shown for `"$EPIBIN/mycommand.exe" arg1 arg2 ...` because DOS interprets the file name line, which is

C:\Program Files\Epiphany, as two *command.com* arguments unless the file name line is placed within quotes.

See Appendix A, “Epiphany Macros” for more information about Epiphany macros.

Note: All system calls are expected to have an exit code of zero, but you can use the Add Job Step dialog box (see Figure 56, on page 193) in EpiCenter Manager to set the *On Error* type to Abort or Ignore, as appropriate.

AGGREGATE BUILDING

Creating aggregates speeds up system performance at the front-end of the Epiphany system. This is because some queries simply require summary data, which can be pre-computed by the system rather than being recomputed for each query. For example, assume that a site has 5,000 products categorized by ten product types. Clarity users may request reports that show data aggregated by product type. If there is a grouping by product type, the Epiphany aggregation program creates a table with ten rows (one for each product type) and pre-calculates “rolled-up” (combined) fact aggregates based on this smaller dimension table. By using significantly smaller fact aggregates, the system can generate reports considerably faster.

When EpiMart is successfully populated, the rows in the fact tables contain foreign keys to dimension table rows. In Epiphany, all dimension primary keys are stored as integers. For example, fact rows may resemble those in fact table Order_0, whereas dimension rows may resemble those shown in Customer_0, Product_0, and Salesperson_0.

Order_0 Fact Table

Customer_key	Product_key	Salesperson_key	Total Sale
6	8	5	\$20
5	6	6	\$30

Customer_0 Dimension Table

Customer_key	Customer Name	Region
5	ABC Company	West
6	Bill's Lighting	West
7	Fred's General Store	East

Aggregate Building

Product_0 Dimension Table

Product_key	Product Name	Platform
6	Product A1	Windows
7	SuperX	Mac
8	Smash-Em	Windows

Salesperson_0 Dimension Table

Salesperson_key	Salesperson Name
5	Gene
6	Sue

All fact and dimension tables that end with **_0** (underscore zero) are called base tables in EpiMart. They contain raw data at the level that is extracted from the source system.

A base dimension is a physical dimension table in EpiMart, such as Customer or Product. Base dimension tables contain the actual dimensional values that are available for reporting or filtering. Attributes, such as Customer Region or Product Platform, can be rolled up. For example, end users often want to create reports of total sales by region. Suppose that Order_0 has hundreds of thousands of records, and Customer_0 has many thousands of records, and that there are only three different regions in the Region field. A report of sales by region against Order_0 and Customer_0 will need to aggregate over thousands of records, causing the report to return answers very slowly. However, a pre-built aggregate with three (rolled-up) rows that already has the total sales by regions could answer the same question very quickly.

After configuring an EpiCenter, the following aggregates might be built:

Order_17

Customer_key	Total Sale
1	\$50

Customer_1

Customer_key	Region
1	West
2	East

Note the following:

- The *Customer_1* table contains only the rolled-up field(s). The number of rows has been reduced so that each row contains a distinct Region. The table name is the same as the base table, with a different number at the end.
- The *Order_17* table contains the same total sales dollar amount, but the number of rows has been reduced (in a larger example, this reduction could be extremely significant). The name of the table is the same as the base table, with a different number at the end.
- The foreign key *Customer_key* has the same name as in the base tables above, but the numbers differ. It is important to recognize that the *Customer_key* field in *Order_17* can be joined only with the *Customer_1* table, since *Order_17* is at the granularity of Regions, not Customers.

See “**Defining Dimension Aggregates**” on page 102 for a description of how tables such as *Customer_1* are built using EpiCenter Manager.

The Aggbuilder Program

The Aggbuilder Program

Note: Aggregate building is performed by Aggbuilder (**agg.exe**), a program similar to EpiChannel. (EpiChannel is the program used to populate your initial EpiMart and to update it on a regular basis; see “Running Jobs: EpiChannel” on page 67.) Aggbuilder uses the same command line, EpiMeta database, and job definition table as EpiChannel. Aggbuilder, however, completely ignores the job steps (that is, system calls and extractors). It uses the job table only for an indication of the *log database* and *working directory*. The structure of Aggbuilder log files is similar to that of EpiChannel except that the *log* directory name is *agg_timestamp* instead of *chn_timestamp*.

The easiest way to invoke aggregate building is via a system call that uses some of the system-call macros to isolate details. To invoke Aggbuilder, add a system call similar to this after the extraction and semantic instances steps, but before any use of the extractor called *End of Extraction*:

```
$$AGG $$EXC_ARGS -j agg_job_name
```

where **agg_job_name** is the name of the job with the proper logging information.

To log to the same places as the current job, use a system call such as:

```
$$AGG $$EXC_ARGS -j $$JOBNAME
```

The Aggregate Building Process

When the Aggbuilder program is executed during an extraction job, the following actions are taken:

Note: All actions occur in the set of mirrored tables (A or B) that is *not* currently active. See “Mirroring: A and B Tables” on page 83 for more information.

- Step 1:** Dimension aggregates are built. Each column set for a base dimension will become a dimension aggregate as long as that column set is included in an aggregate group via EpiCenter Manager’s Aggregate Group dialog box (see “Aggregation and Aggregate Grouping” on page 115). Indexes are built for these tables as appropriate.
- Step 2:** For each fact table, all enabled aggregate groups to which that fact belongs are examined. Aggbuilder determines all possible combinations to be built based on the dimension role definitions and creates unique table names. Fact aggregates are physically built and indexed.
- Step 3:** Aggbuilder records what it has built in a set of read-only metadata tables in EpiMeta. You may use EpiCenter Manager to browse these lists. The Epiphany Application Server uses these same lists at runtime to decide which aggregates to use to satisfy an end-user query.
- Step 4:** All activity is logged to Epiphany’s logging tables.

MOMENTUMBUILDER

MomentumBuilder is the executable that creates special Momentum tables, such as mini-dimension tables and clusters, which are described in the discussion on Momentum in Chapter 3. The order of the extraction for a Momentum EpiCenter is as follows:

Step 1: SQL/Semantics

Step 2: Aggbuilder

Step 3: MomentumBuilder

Step 4: End of Extraction

In a Momentum EpiCenter, **EpiMgr** creates a default MomentumBuilder job step that runs the MomentumBuilder program (**MomentumBuilder.exe**) with the appropriate arguments. Similar to Aggbuilder's **agg.exe**, **MomentumBuilder.exe** is usually run as a system call from an EpiCenter Manager job. You may also invoke it from the command line (for example, when you run MomentumBuilder as the sole extraction process).

The **MomentumBuilder.exe** command-line arguments are as follows:

```
-h
-?
-INSTANCEDIR
-EPIBINDIR
-S server_name
-D db_name
-U username
-P password
-v verbosity level
```

A given Epiphany query uses only a *single* fact table (either base or aggregate). Typically, as a user drills down into a result set, the physical table that services each query moves from a high-level aggregate (with few rows) to a larger aggregate (possibly using an index on the aggregate). For the user's response time to be acceptable in each of these cases, you must properly configure both aggregates (as described in the previous section) and custom indexes.

All aggregates are indexed in the same manner as the base table (insofar as the aggregate has the same dimensionality as the index). For example, if the base table has an index on *customer_key*, *product_key*, and *territory_key*, then an aggregate that contains only the Customer and Territory dimensions will have an index on *customer_key* and *territory_key*.

Note: If two different indexes on the base table result in the same index on the aggregate, then the aggregate index is built only once.

By default, each fact table has one index built on its dimensional keys during extraction (on Microsoft SQL Server this index is also the CLUSTERED index). For all fact tables, the leading term of the index is *date_key* because a wide class of queries filters on the time dimension. For example, if a system contains data for multiple years and a user asks for a single calendar month, the database almost certainly will use an index when it services this query (whether via the base table or an aggregate). All other dimension roles in the fact table are also included in this index. The order in which they appear is determined by their order in the constellation within EpiCenter Manager's directory tree. To re-arrange this order, right-click a dimension role folder, and select Up or Down from the pop-up menu.

A custom index is the metadata definition of indexes to build on fact tables in EpiMart. Normally, each fact table is given a single index. You can use EpiCenter Manager (the Custom Index tab in the Fact Table dialog box) to:

- configure all other indexes on the fact table.
- assign each index a logical name, although the name is not used when building the physical index.
- choose the dimension roles that make up the index, and their order.

Note: You will use the Custom Fact Index semantic type to actually build the indexes. (See Appendix F, "Semantic Types.")

The UNKNOWN Dimension Row

It often makes sense to build several different custom indexes, each with a different dimension role as the leading term. If, for instance, end users often filter on attributes of the Customer and Product dimensions, then you could build a custom index on the combination (*customer_key*, *product_key*) and another on *product_key* by itself. There is no need for the second index to also include *customer_key* because the first index could easily service joint customer/product queries.

THE UNKNOWN DIMENSION ROW

When a fact staging table is loaded during an extraction, its dimension source system key columns (columns whose names end with *sskey*) normally refer to rows in the base dimension staging tables. (The column name being referred to in the base dimension staging table also ends with *sskey*.) These source system keys are the actual values used to form the relationships on the source system between the fact and dimension source tables.

Some source systems do not enforce referential integrity, however, which may result in source system fact tables that have “dangling references,” or references to non-existent dimension values. Another possible source of a dangling reference is an optional or null foreign key in the source system. In EpiMart, all fact dimension keys must point to valid base dimension entries; and there should be no dangling references.

To prevent dangling references, each base dimension in EpiMart is populated automatically with a single special row called the UNKNOWN row. This row is always available for mapping fact dimension keys that would otherwise not point to a valid base dimension row.

where:

- h and -?** Display on-line help.
- INSTANCEDIR** The path to the root directory of the Epiphany instance where MomentumBuilder will run.
- EPIBINDIR** The path to the **win32** directory in the instance (this is the directory provided by the EPIBIN macro).

Either INSTANCEDIR or EPIBINDIR must be specified so that **MomentumBuilder.exe** can locate the **classes** directory in which the MomentumBuilder Java file has been placed by the installer.
- S** Database server name.
- D** Database name.
- U** Database user name.
- P** (Optional) Database password. If omitted, uses a null password.
- v** (Optional) Verbosity level.
- NOMIRROR** (Optional) Runs MomentumBuilder on the currently active datamart.

The default MomentumBuilder job created by the EpiCenter initialization SQL runs **MomentumBuilder.exe** using the EPIBINDIR argument, which is available under Remote Administration installs where there is no *instance_name*.

Verifying MomentumBuilder Extraction

Alternatively, on the command line you may specify **-I** followed by an instance name. In this case, **MomentumBuilder.exe** retrieves all necessary directory locations and database login information from that instance's key in the Registry. This approach is not used by default because it will fail in Remote Administration installs in which the Instances Registry key is not present.

Verifying MomentumBuilder Extraction

To determine if MomentumBuilder extracted the appropriate number of individual and groups, you need to figure out how many rows there should be in the Momentum tables. The tables *all_groups_* and *all_individuals_* are derived from *ind_group_joiner*.

- *all_groups_* contains the real keys that correspond to all of the distinct *group_key* values in *ind_group_joiner*.
- *all_individuals_* contains the real keys that correspond to all of the distinct *indiv_key* values in *ind_group_joiner*.

The size of the *all_individuals_* table should equal the number of rows in *ind_group_joiner* where the *indiv_key* is not equal to 1.

Please refer back to this section after you have read the Momentum-related material in Chapter 3.

CUSTOM FACT INDEXING

EpiCenter supports two methods that dramatically improve query performance: indexing and aggregation. Each accelerates different classes of queries.

In general, the presence of aggregate tables accelerates queries that answer high-level business questions, such as Sales by Business Unit and Fiscal Year. Deep drill-down queries, or highly filtered queries that ask for a small subset of data, are typically enhanced by an index. A fact base table is usually the largest table in EpiMart. Even tables with millions of rows can be accessed quickly via an index when only a small number of data pages is needed to service a query.

Referring to the UNKNOWN Row during Extraction

When writing fact staging SQL statements, you can refer explicitly to this special row by populating the *sskey* dimension role column with the special string UNKNOWN. For instance, on a Microsoft SQL Server source system, you might use the ISNULL() function as follows to translate all null values to UNKNOWN:

```
SELECT
    ...
    ISNULL(cust_code, 'UNKNOWN') customer_sskey
    ...
FROM
    ...
```

Referring to the UNKNOWN Row during Extraction

For a base dimension such as Customer that has dimension columns *customer_name* and *region*, the UNKNOWN row has actual values similar to any other row. By default, each dimension column assumes the literal value UNKNOWN for each of its textual attributes. However, you may use EpiCenter Manager to override this default value so that end-user queries return a different value. To do so, enter a literal string (without quotes) in the Dimension Column dialog box (accessible from the General tab of the Base Dimension dialog box shown in Figure 16, on page 100). Your string will be used instead of UNKNOWN.

The CountUnjoined Semantic Type

While extracting a fact staging table, one often does not know whether the values that appear in the dimension foreign key columns actually refer to dimension rows. While the technique described above for converting null values to the UNKNOWN string works for missing dimension foreign key values, it will not work when a dangling value appears. If no action is taken, then these fact rows will be lost during extraction since the first step of most fact semantic types involves an inner join between the fact staging table and the dimension mapping tables (see Appendix F, “Semantic Types”).

Normal Extraction Order

The CountUnjoined fact semantic type is specifically geared towards converting these dangling fact references to the special UNKNOWN value. Scheduling this semantic type (after the fact staging table has been loaded) during an extraction will cause a scan of the fact staging table with joins to each of the dimension mapping tables. All *sskey*'s in the fact staging table that do not map to extracted dimension rows will be converted to UNKNOWN.

Note: Execute the CountUnjoined fact semantic type before executing any other semantic types on this fact staging table.

In general, it makes sense to execute semantic types for base dimensions before any of the semantic types for facts (including CountUnjoined) because the base dimension semantic types produce the mapping tables needed by fact semantic types (see "Normal Extraction Order" below).

Normal Extraction Order

Epiphany's recommended sequence of events during an extraction is as follows:

- Step 1:** Load fact staging tables.
- Step 2:** Load base dimension staging tables.
- Step 3:** Execute base dimension semantic Instances.
- Step 4:** Execute fact semantic instances.

The reason for loading the fact staging tables before base dimension tables is subtle. If Epiphany is extracting from a live source system, then new dimension and fact rows could be created while the extraction occurs. If dimensions were extracted first, then when facts are extracted, a new dimension row (such as a new Customer) might have been created, but was missed. If a fact that refers to that new Customer is then extracted, the fact row shows up in the Epiphany system with an UNKNOWN Customer value (because that Customer row was missed by the extraction). By extracting the facts first, the system might not receive the fact row in this extraction, but will retrieve it in the next extraction.

RUNNING JOBS: EPICHANNEL

EpiChannel is the Epiphany extraction program (**extract**) that you will run to populate your initial EpiMart and to update it on a regular basis.

EpiChannel first opens an ODBC connection in the source system and then opens a database library connection to the target system and initializes tables. It establishes start and stop date time boundaries and replaces certain special strings. After replacing these strings, EpiChannel executes a user-defined sequence of SQL statements or system calls.

The SQL is pre-processed to match warehouse and database vendor syntax. System calls are pre-processed to supply directory locations and database login information.

Before attempting the first job step, EpiChannel verifies the availability of databases and tables (if these options are selected in EpiCenter Manager's Job dialog box) and expands most macros. If these fail, the job does not run.

The job will halt when it encounters the first error in any step, unless that step has an error code set to print or ignore the error instead of to abort it. Error codes are set through EpiCenter Manager. You will use the SQL Statement dialog box (Figure 52, on page 185) for extractors, and the Job dialog box (Figure 54, on page 190) for system calls.

Whenever a job succeeds or fails, e-mail notification is automatically sent to the database administrator (or any designated list of e-mail recipients). See "Configuring E-Mail" on page 196 for more information.

The EpiChannel Command Line

This section describes the command-line syntax you will use to invoke EpiChannel at the console's DOS prompt. These command-line parameters indicate the database with the job information (that you configured using EpiCenter Manager) and the parameters related to interactive debugging. Command-line parameters may also specify the initial debugging level, the maximum selection limit, or that the jobs are to be trial runs.

The EpiChannel Command Line

To invoke EpiChannel, enter the **extract** command on the console's DOS command line followed by the parameters described below.

Note: EpiChannel inherits the case sensitivity of the database engine.

```
extract      -h  
              -?  
              -v integer_verbosity_level  
              -I instance_name  
              -S server_name  
              -D db_name  
              -B db_vendor_name  
              -U username  
              -P password  
              -t trial run  
              # row_count  
              -j job_name
```

where:

-h and -? Display detailed on-line help.

-v *integer_verbosity_level*

Runs the EpiChannel job in verbose mode, which displays the SQL on the screen. See "EpiChannel Debugging Levels" on page 78 for more information.

-I *instance_name* Specifies which Registry instance key to use. This parameter is mutually exclusive with many of the other parameters because it causes them to be read from the Registry.

-S *server_name* Specifies the name of the server where the EpiMart resides.

-D *db_name* Specifies the name of the EpiMeta database.

EpiChannel Registry Keys

- B *db_vendor name*** Specifies the name of the EpiMart database vendor.
- u *username*** Specifies the username for the EpiMeta database.
- P *password*** Specifies the password for the user of the EpiMeta database.
- t** For testing purposes, this is the trial-run option that simulates execution of the SQL on the source and destination databases.
- # *row_count*** Specifies the maximum number of rows to fetch on any SQL statement.
- j *job_name*** Specifies the name of the Job as defined in the EpiMeta.

EpiChannel Registry Keys

The EpiChannel program uses the Registry keys shown in EpiChannel Registry Keys, which are located in the following Registry path on the local machine: **Software\Epiphany\Instances***instance_name*.

In the table below *Default Instance* is the string value for Default in the Instances Registry key under the Epiphany Registry key.

Table 2: EpiChannel Registry Keys

Key	Value
1	Default Instance
2	[Default Instance]/DBVendor
3	[Default Instance]/Database
5	[Default Instance]/Server
6	[Default Instance]/Username
7	[Default Instance]/Password

Output Files

By default, EpiChannel reads the Default Instance key and uses its value to open the Registry tree that holds all initialization parameters. You can override this by specifying a different name using the **-I *instance_name*** option.

For example:

```
extract -I name
```

forces EpiChannel to read all Registry values from **HKEY_LOCAL_MACHINE\...\Epiphany\Instances*instance_name***.

Other EpiChannel command examples follow:

- A command (in a system call) in which all Registry variables are set:
extract -j job1
- A command example in which no Registry variables are set:
\$\$EXC_CMD

Output Files

Jobs may log to files or databases, and these logs can be directed to multiple destinations simultaneously. For example, one log could be placed on the local machine and another on a network server.

The execution of jobs is logged as a unit. Statistics are collected for job steps and for entire jobs. The location of the log is established by associating data stores with the *log* or *working directory* roles for a job, for which you will use the Job dialog box in EpiCenter Manager (see Figure 54, on page 190).

Extracting New Rows Only

An efficient extraction process pulls only data from the source system that has changed or been added since the previous extraction. The way that the extraction process recognizes this subset of data depends on how the source system identifies changes in data.

Epiphany supports the following methods of identifying changed data:

- **Date or Date/Time fields in source rows**
A source row, or a table it joins to, contains a date or date/time indication of when it was inserted or last updated.
- **Timestamp fields in source rows (Microsoft SQL Server only)**
Values can be “stamped” in some way to indicate their status. Epiphany provides Microsoft SQL Server timestamp fields for source rows. A source row, or a table it joins to, has a column that contains Microsoft SQL Server data type *timestamp*. The database engine automatically updates this column whenever the row is inserted or updated. This field does not actually contain any record of the time that the update occurred, but rather values that always increment, such that any row edited after another row will have a higher timestamp.
- **Highest numeric or string value in named columns in source rows**
A particular column in a particular table is presumed to contain a value that is always increasing from row to row; for example, Order Number or Policy ID. For this method to be useful as a subset selection criterion, however, either the rows must never be updated, or else the column must be changed by the source system whenever an update occurs.

How EpiChannel Identifies Data To Be Extracted

How EpiChannel Identifies Data To Be Extracted

The following topics describe how EpiChannel identifies which subset of data to extract:

- How EpiChannel reads values

EpiChannel attempts to get a consistent set of dates/timestamps/column values so that the same single moment in time is captured by the WHERE clauses generated by the extraction set identification macros. The extraction program minimizes the chance for changes to occur to one set of values, and not to another one, by reading all sets one after another with no intermediate actions. While this gives highly consistent values, it is possible that a change may occur between the time EpiChannel reads the first table/column and when it reads the last one, especially if multiple databases are involved.

The values read are stored as the “last” values only if the job commits its work. If any system call or SQL statement has an error that aborts the job, it is as if the value stamps had never been read or modified. In the unlikely event that a job aborts during the commit phase, the timestamps might not be synchronized with each other.

The date and timestamp values are read from a table-independent current value such as SYSDATE or TO_EPIDATE.

- Extraction subset extraction

Epiphany supports *extraction subset extraction* by reading and recording the current values of these fields at the start of the run and remembering these values for the next run. The values are made available in SQL through the SQL macros described in Appendix A, “Epiphany Macros.”

- Ignore Timestamps option

The Ignore Timestamps option, which you can set using EpiCenter Manager’s Job dialog box, causes all of the value range expressions to become (1=1), or true, which means that all data is selected without regard to its value range.

- “Priming” value range memory

The value range memory needs to be “primed” by a trial run whenever you add a new range expression. Basically, the first run causes the memory to start tracking the values for the next run, but all ranges are 1=1 for the current run. The second run causes the “current” values to be read, but all ranges are 1=1 because there are no last values. The third and successive runs have both current and last values, and the range expressions are valid. If you have changed the SQL and are in doubt, make a trial run, which will prime the value memory with any new value expressions.

- Removing and Re-adding Extraction Set Identification Macros

The presence of an extraction set identification macro in an enabled SQL statement causes the date, timestamp, or column values referenced in that expression to be read at the start of the extraction job and “memorized” if the job is successful. It does not matter if the macro is in a SQL comment, or even within another Epiphany macro such as \$\$ORACLE[]. The macro is processed (and the values read) without awareness as to whether the proper parsing of the greater SQL statement indicates the expression is enabled.

Once a value is memorized, it stays memorized. If you remove the extraction set identification macro for some runs and later re-add it, the values memorized when the expression was last used will be the “start” range of the re-added macros. If you want to “forget” the previous values, open the extractor’s dialog box in EpiCenter Manager, select the Filters tab, and clear or delete the memorized value.

Memorized values are shared between expressions within an extractor. For example, if two extraction set identification macros reference the value of *current_orders.order_number*, then that value will be read and memorized just once. If one of these macros is removed, the other continues to read and increment the maximum *order_number*. If the removed macro is then re-added, it will expand with the incremented *order_number* value. If all extraction set identification macros that reference *current_orders.order_number* are removed, then the lookup and memorization of *order_number* will not occur. The next macro to reference that *order_number* will use the last value

EpiChannel Debugging

memorized before the macros were removed. If you do not want to share the memorized value between SQL expressions, place the SQL expressions in different extractors.

Note the following:

All dates use the same memorized value, and all timestamps use the same memorized value. Thus date continues to be incremented if any extraction set identification macro referencing date remains. Similarly, the timestamp increments as long as any macro referencing timestamp remains.

The extraction set identification range memory is shared between jobs. Two jobs can increment values that both jobs use. Depending on the circumstances, this may or may not be desirable.

- Memory for last values

The “memory” for last values is associated with an extractor and not a SQL statement or a job. If the same extractor is used in different jobs, the jobs will pick up each other’s value updates. For example, the Weekend job will see the values updated by the Weekday jobs if they share extractors.

If the same SQL step is used in two extractors, it will have different “last” values for each extractor, which is desirable because the extractors may access different databases that have different highest values and times.

EPICHANNEL DEBUGGING

The EpiChannel debugging mechanism is thorough and well annotated. If you set a high enough verbosity level, the complete anatomy of the job is revealed in real time at the console, or historically in the log file. The EpiChannel **-v** option controls the verbosity level. (Running **extract** with the help (**-?** or **-h**) option lists and defines these verbosity levels.)

Job Output

You may display EpiChannel output for debugging purposes on a screen or direct it to a file. The text is displayed on the screen in fixed fonts, such as Courier. The number of characters that can be displayed per line is determined by the number of columns. (You may use EpiCenter Manager's Job dialog box to set the job log width to 80 or 132.) The output is designed to be adequately displayed on low-end monitors.

At the top of the debugging screen/log, you will find the names and directory locations of the job's EpiMeta database, as well as its *job log*, *log file*, and *log database*.

Typically, the job begins with the truncation (deletion) of old staging tables and the clearing of indexes. The processing of each step is displayed sequentially; for example:

1.1 Processing step 'PreMfg' as an extractor

followed by the extractor's source and destination databases. Steps that are not enabled are numbered, but not shown.

Each operation follows on a separate line. You can identify the type of operation by the letters or symbol(s) that precedes it:

- **St** = A set of SQL statements. The members of this group can be either stand-alone SQL, extraction statements (pull/push), or semantic instances (data merges). Steps may be one of these types:
 - **< =** Stand-alone SQL applied to the source data store.
 - **> =** Stand-alone SQL applied to the destination data store.
 - **<> =** Pull/push SQL that transfers data from the source data store to the output data store.
- **SI** = Semantic instances.
- **Sy** = System calls.

Job Output

The EpiChannel output provides summary information that is based on the statement type. The summary information is prioritized and displayed in columns to the right of the operation. If there is not enough space, only the highest priority information is given.

For SQL pull/push statements, the output includes the time taken for the pull/push, the number of rows in the table before the operation, the number of rows transferred, and the number of rows in the table at the end of the procedure. If the log file's width is 132, the number of bytes is also shown.

Stand-alone SQL statements have time as their only metric. Semantic instances define their own metrics.

To conserve space, the summary information is labeled by two-digit codes (tokens) that appear to the right of their numeric values. These tokens are defined as follows:

- **Ti** = Time taken, rounded to the nearest second.
- **St** = Rows present in the table at the start.
- **Ex** = Rows extracted.
- **En** = Rows present in the table at the end.
- **UN** = Rows unjoined. These fact table rows contained references to dimension values not present in the EpiMart or in the newly extracted data.
- **PR** = Rows processed; a count of the gross number of rows examined.
- **IN** = Rows inserted; a count of the rows that were successfully moved into the EpiMart. Rows are sometimes omitted because they are earlier than previously recorded data, or because they are redundant to data present elsewhere in the same staging table.
- **MO** = Rows modified. Modifications typically adjust references to missing dimension values such that they become references to the predefined UNKNOWN dimension value.

For example, the lines:

```
Statement NameTime Metric1 Metric2 Metric3
```

```
St wb bump the test day
```

```
< wb bump the test day
```

```
<> appl real          0 St  3 Ex  3 En
```

```
SI Product  1 TI  3 IN  0 MO  3 PR
```

indicate the following:

- A set of statements called `wb bump the test day` was called.
- There is no summary information for this set.
- An execute-only SQL statement called `wb bump the test day` was run against the source system and took less than a half second to execute, and therefore has no time statistic.
- An extraction statement called `appl real` inserted 3 rows into a previously empty table and took less than half a second to execute.
- A semantic instance called `Product` took about a second to execute and processed 3 rows, modifying the contents of none, and inserting 3 rows as data into the EpiMart.

The summary table at the end of the report shows totals for the amount of time each operation took and the number of rows, bytes, and metadata objects that were extracted. Using this information, you can quickly determine if a problem resides in EpiChannel, the semantic instances, or in the extraction SQL statements or system calls.

Error Messages

Error Messages

An EpiChannel error message distinguishes between externally generated errors, such as database errors, and internally created messages, such as a file data store that references a nonexistent directory. Each error includes information about the database/file that caused the error, and which specific part of the code detected the error. Most errors contain information about the operations affected or aborted by the error, the metadata objects that specified these operations, and the program's plan of revised actions. Many errors also suggest corrective action.

There are specific error messages for the following:

- SELECT statements that do not match columns in the metadata. These display mappings indicate which return values were or were not accepted.
- Rows that fail to insert have their complete set of data displayed.

EpiChannel Debugging Levels

EpiChannel's verbose (`-v`) option allows you to select how much of the SQL displays on the screen during extraction. For a description of each debugging level, run EpiChannel with the `-h` or `-?` parameter. The default level shows jobs, steps, SQL statements, semantic instances, and statistics.

The lowest level shows only jobs and errors. Increasing levels show the SQL pre- and post-expansion of macros, templates, and blocks within semantic instances, or even each individual row returned from a SELECT statement. At the highest levels of verbosity, execution pauses between SQL statements and between returned rows, which allows you to trace behavior precisely.

You may even use EpiChannel interactively to alter the data fetched by a SELECT statement before the row is forwarded for insertion. See "Setting Breakpoints" on page 79.

Note: To turn off verbosity, select `-4`.

Setting Breakpoints

You may set breakpoints on SQL statements that change the debug level *before* the SQL is issued. These breaks can also be associated with particular rows in the return set. For example, it is possible to set a breakpoint just before row 3021 of the *Orders table extraction* SQL statement.

You can use the EpiCenter Manager's SQL Statement dialog box to set breakpoints (see Figure 52, on page 185). Follow these steps:

- Step 1:** Select the debug level. These levels correspond to the verbosity levels on the **extract** command line (or in the Execute Job dialog box shown in Figure 55, on page 192).
- Step 2:** Select the row number at which the debug level you selected above will go into effect; this is the row number immediately *before* the SQL statement is executed. If the row number is not empty, and the level is "row pause" or higher, the pull/push loop stops fetching before inserting that particular row and waits for permission to proceed. You may alter the data during this pause.

When you set a breakpoint, it stays set for the remainder of the job. This prevents you from having to set a debug level on entire groups, but it does necessitate your setting the breakpoint back to the original value later in the job flow if you do not want the entire job debugged (and remembering to remove this breakpoint).

All SQL issued against the source, destination, or metadata databases will be logged, if specified by the debug levels.

Trial Runs

You can run EpiChannel in trial-run mode, in which SQL is not actually issued against the source or destination systems. This option is recommended for testing the structure of an extraction job.

EpiChannel Output

Maximum limits can be set on the number of rows to be fetched from any SELECT statement. When used, these limits essentially cause the SQL to be checked for syntax on a small selection set. The execution time for all SQL executed is tracked and logged.

EPICHANNEL OUTPUT

You must provide EpiChannel with one and only one file data store with the role of *working directory*. A working directory is a log directory that is also used as the location for temporary files associated with the job.

You may, however, have any number of file or database data stores that have the role of *log*. The locations referenced by data stores with these roles are known as log directories and log databases, depending on their data store type. You may use more than one log database and more than one log file. In this case, all logging activity is duplicated to all of the listed data stores. Having a backup of each kind of data store is recommended if you require logging information to be constantly available for the administration of your system, or if data stores could possibly go off-line or be destroyed.

You may also use multiple listings so that one log is local to the host and one is remote. For example, during installation and early testing, you might direct one database log to a database hosted by Epiphany for the monitoring of your site's activities.

Note: The EpiCenter Manager default data stores may be sufficient for your site.

Log and Working Directories

The location specified for a *log* or *working directory* is a parent directory under which a new subdirectory will be created for every job. The subdirectories are named based on a timestamp from the start of the EpiChannel run. If there is already a subdirectory of that name, then a suffix is added to make the name unique to that directory. Under the subdirectory is a file called *jobname.log* with the log and informational messages from that job.

Log Files versus Log Databases

EpiChannel can log to both files and databases. Log files provide detailed information regarding a single job and log databases provide complementary but less detailed status and performance information about all jobs. Log files help you to determine the success of a job and suggest corrective action. Log databases enable you to analyze the behavior of jobs over time in order to detect troublesome areas, or areas of degrading performance.

Note: A log database is not a log device. A log device is a construct in SQL Server that logs database transactional activity.

MONITORING JOBS

You may use the Windows NT Performance Monitor to determine the current status of the database extraction process; for example, you can find out if the cause of a longer than normal extraction process is the result of network performance problems, or simply more rows being extracted and processed.

You may use one or more of these measures to monitor job behavior:

- The total number of jobs started. Use this to see transitional boundaries in the work being done.
- The total number of extraction nodes started. Use this to see transitional boundaries in the work being done.
- The total number of rows transferred in a pull/push operation. This value is reset with each extraction statement.
- The total number of bytes transferred in a pull/push operation. This value is reset with each extraction statement.
- The number of rows committed. This value is reset with each extraction statement.
- The rate of row transfer in a pull/push operation (rows per second).
- The rate of byte transfer in a pull/push operation (bytes per second).

Running the Performance Monitor

- Total number of semantic instance blocks called. Use this to see transitional boundaries in the work being done.
- The rate at which semantic instance blocks are called. This shows the percentage of the clock time spent in semantic instance calls.
- The number of meta objects in the job, which is a measure of the complexity of the workflow.

You may combine these metrics with the other measures available through the Performance Monitor. For example, the bytes per second can be displayed along with the TCP packets per second, the average length of the disk queue, and SQL Server-specific measures.

Running the Performance Monitor

See the *Epiphany Installation Guide* for instructions on how to set up the Performance Monitor for EpiChannel usage.

After setting up the Performance Monitor, you can monitor any EpiChannel jobs that are run on this machine. Follow these steps:

- Step 1:** Start the NT Performance Monitor. (It is located in the *Start\ Programs\Administrative Tools* menu on most machines.)
- Step 2:** Click the ADD or plus (+) button in the Performance Monitors' window.
- Step 3:** Select Epiphany Channels from the object list.
- Step 4:** Select the measures you want to monitor from the counter list.

MIRRORING: A AND B TABLES

The Epiphany system uses the metadata definitions of facts and dimensions to construct the physical EpiMart tables. Table naming conventions indicate how the table is used. For instance, if a base dimension table named Customer has been defined via EpiCenter Manager, the system constructs these tables:

- Customer_0_A
- Customer_0_B
- CustomerMap_A
- CustomerMap_B
- CustomerStage

The *_0*, *Map*, or *Stage* suffixes are needed for the normal operation of EpiMart. Note that the *Customer_0* and *Map* tables appear twice in this list: once each with a suffix of A and B. The purpose of this table “mirroring” is to allow extractions to occur without disrupting the live usage of a system.

At any time, either the letter A or B is “active,” meaning that all end-user queries will operate against tables with that suffix. This setting is determined by an option in the EpiCenter Manager’s Configuration dialog box (described in Appendix B, “EpiCenter Configuration.”) called *current_datamart*. If this option is set to A, then all end-user queries will be routed against tables that end in A (and all of the B tables will lie dormant). The B tables contain data that is one extraction older than the A tables.

During an extraction, the data in the active tables (A in this example) is never changed. Instead, the B tables are constructed by combining the A table data with any newly extracted data from the staging tables. In other words, the dormant set of tables is rebuilt every extraction with the latest data. Not only does extraction of the base tables (those that end with *_0*) occur in the dormant set, but aggregation is also performed on the B tables. In this way, end users can continue to query against the A tables, without knowing that extraction and aggregation are occurring simultaneously against the B tables.

SQL Limits for Facts

If extraction finishes successfully, then the value of the configuration setting *current_datamart* is toggled to be the dormant letter (B in this case). As soon as the Epiphany Application Server (described in Chapter 4) is notified that the data extraction was successful, all end-user queries will be directed against the B tables. In this way, the A tables, which contain one less extraction's worth of data, become the dormant set. The next time EpiChannel is run, the A tables will be recreated, and the letter A will once again become active.

Note the following:

- Aggregate building is always performed against the set of tables not equal to the *current_datamart* setting.
- In order to completely re-extract an entire system, the set of tables whose suffixes equal the *current_datamart* setting must be truncated. (This, however, affects current end users of the system and will be resolved by the introduction of a new semantic type that allows complete re-extraction without truncating any EpiMart tables.)

SQL LIMITS FOR FACTS

When configuring your datamart, observe these SQL limits, as detailed in the SQL Server documentation:

- MONEY facts are limited to +/- 922,337,203,685,477.5807
- QUANTITY facts are limited to +/- 999,999,999.99
- INT facts are limited to +/- 2,147,483,647

The selection of a datatype that is insufficiently large enough to store your data may affect the accuracy of Epiphany software extractions and queries.

CHAPTER 3

EPICENTER MANAGER

EpiCenter Manager is Epiphany's versatile program for configuring, administering, re-configuring, and updating your database. You will use its graphical user interface to define your EpiMeta database table schema, configure ticksheets for end users, and set up the jobs needed for data extraction. You will also use EpiCenter Manager to perform administrative tasks, such as restricting access to ticksheets and sending e-mail notification of job status. When your schema changes incrementally, EpiCenter Manager allows you to update it "on the fly," without rebuilding the entire EpiMart.

This chapter describes how to use EpiCenter Manager to set up your databases for optimal extraction of your source data, to define jobs, and to administer your system. Instructions are also given for configuring Clarity, Relevance, and Momentum ticksheets.

PLANNING YOUR EPICENTERS

EpiCenter Manager is a program for experienced database administrators who are familiar with their site's database and database needs. Because incorrectly updating schema can corrupt an existing database, it is important that you understand Epiphany system concepts and how to use EpiCenter Manager in a way that is appropriate for your site. Before using EpiCenter Manager, be sure to read Chapters 1 and 2 of this *Guide*, which describe the Epiphany system and the database extraction process. Many of the key terms, such as EpiCenter, EpiMeta, and EpiMart are described in these chapters.

Before you configure an EpiCenter, your site needs to analyze its business model to determine:

- How many EpiCenters are appropriate.

An EpiCenter is a construct for organizing your metadata into a database called EpiMeta that defines a single EpiMart database. EpiMart is where the data extracted from your source systems resides. If your system has adequate memory and storage (see the *Epiphany Installation Guide*), and your business model warrants it, you may set up multiple EpiCenters. For example, you may have one for development and one for production.

- The number of constellations within each EpiCenter.

This includes the fact tables and dimension tables (and aggregates) that should be logically organized into a constellation. Although every EpiCenter has at least one constellation, most EpiCenters will have multiple constellations. The number depends on the site's business model; for example, one for each business process.

There is a separate constellation especially for the Momentum application. Momentum also has associated constellations, called Momentum Adjacent constellations. When you are working with Momentum, you will more than likely create adjacent constellations specifically for Momentum demographic queries.

- The kinds of Clarity, Momentum, and Relevance ticksheets you will configure for end users.

- The extraction process for your site.
Analyze the kinds of jobs and job steps needed; for example, you may want to create job flow chart(s). After you design the system at this higher level, you will be better able to—
 - Identify the extractors (SQL statements and semantic instances) needed to extract information from the source system(s) and write these extractors.
 - Define external tables.
 - Identify requisite system-call commands.

Note: For security purposes, two functions of EpiCenter Manager are available as separate programs: Web Builder (for creating ticksheets) and Security Manager (for assigning access rights to log onto the system and to access, modify, and save ticksheets).

GETTING STARTED

Follow these steps to start EpiCenter Manager and to register a database server:

Step 1: Complete the installation as described in the *Epiphany Installation Guide*.

The installation procedure includes creating new, empty databases on the server for the EpiMeta and EpiMart databases and installing the Epiphany software. Although you create two databases, EpiMeta is the one you work with using EpiCenter Manager. EpiMeta defines the database tables for your data warehouse (the EpiMart).

Step 2: Start EpiCenter Manager.

During installation this program executable (**EpiMgr.exe**) is placed in the Epiphany directory by default (**C:\Program Files\Epiphany**) and can be selected from the Start menu: *Programs\Epiphany\instance_name\EpiCenter Manager*.

The EpiCenter Enterprise Manager window is displayed.

Getting Started

At the top level is the Servers icon. Follow these steps to register your server:

Step 1: Choose Register from the Server menu. The Server Properties dialog box (Figure 9, on page 88) is displayed in the window.

Step 2: Select the Server Type from the drop-down list and enter the database server's name, and the username and password for this server.

Your database server machine icon and the EpiCenters folder (or directory) appear in the hierarchical tree structure (see Figure 10, on page 89). This server icon represents the server where the EpiMart and EpiMeta databases for this EpiCenter reside.

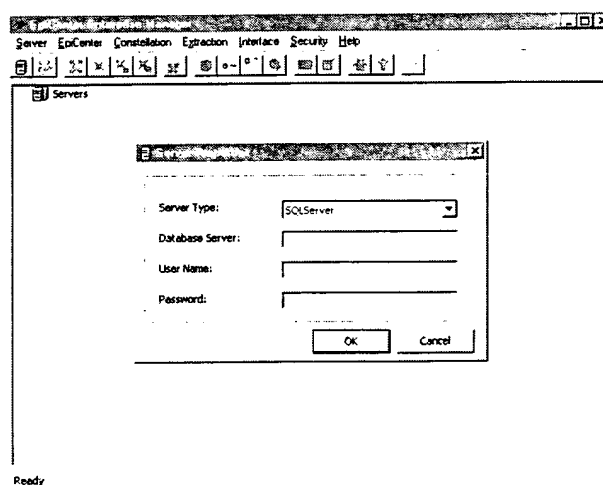


Figure 9: Registering the Database Server

Step 3: Click Cancel to close the empty Choose EpiCenter dialog box.

Later you may use this dialog box to select and open an already initialized EpiCenter as described in "Working with an Existing EpiCenter" on page 92.

When you register a server for the first time, after connecting to that machine, EpiCenter Manager displays the Register EpiCenter dialog box. Close this dialog to continue with the initialization.

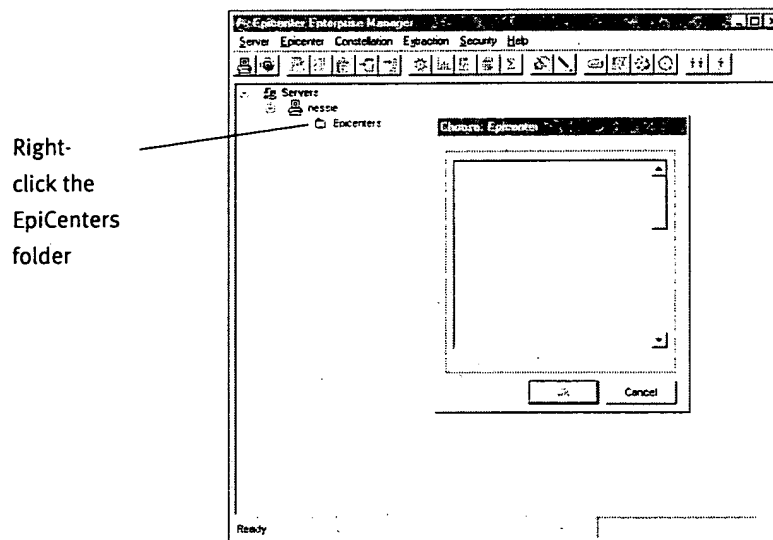


Figure 10: EpiCenter Enterprise Manager Window

Follow these steps to initialize your EpiCenter:

- Step 1:** Right-click the EpiCenters folder in the EpiCenter Manager Enterprise window (see Figure 10) and select Initialize EpiCenter from the pop-up menu. The Initialize EpiCenter dialog box is displayed (see Figure 11, on page 90).
- Step 2:** In the EpiMeta Name drop-down list, select the name of the new, empty database for the metadata that you created during installation (as described in the *Epiphany Installation Guide*).
- Step 3:** In the EpiMart Name drop-down list, select the name of the new, empty database for your customer data that you created during installation.

Getting Started

- Step 4:** The basic initialization is for Clarity only. Select Include Relevance and Include Momentum if you have installed these products.
- Step 5:** Enter the name of the directory for the default job log file. The default location is the Windows **TEMP** file location on the current machine.
- Step 6:** Click Build to start building a generic EpiMeta database (as specified in the *Epiphany Installation Guide*). The status of the build is displayed in the lower part of the (now expanded) Initialize EpiCenter dialog box.

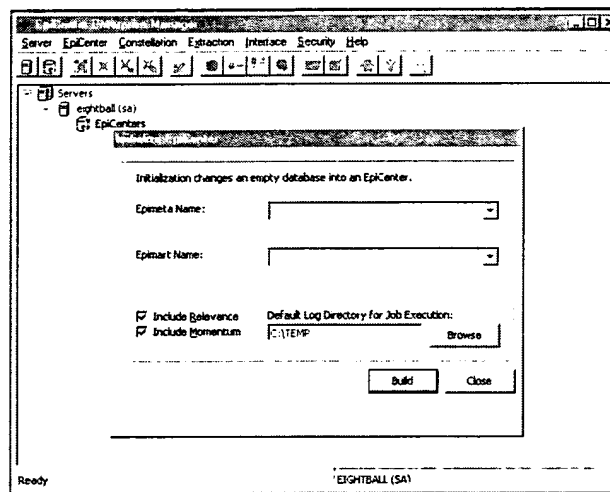


Figure 11: Initializing Your EpiCenter

The new EpiCenter icon is added to the directory tree with folders for Base Dimensions, Constellations, Extraction, Shared Interface, and Security (see Figure 12). Note that the icons in the window are arranged in an Explorer-like hierarchical tree.

Populating the Date Dimension

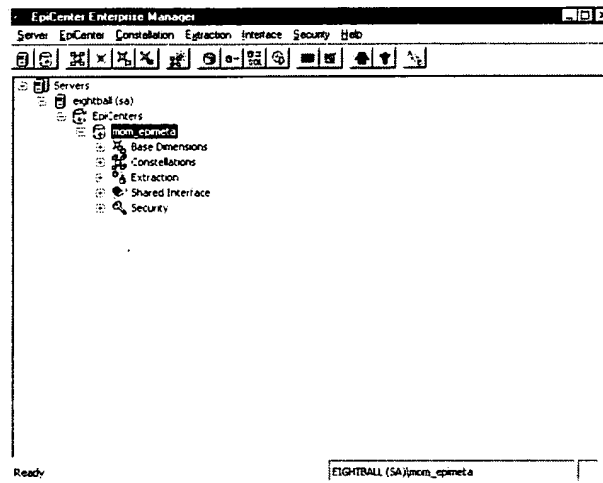


Figure 12: EpiCenter Enterprise Manager Window

POPULATING THE DATE DIMENSION

The date dimension table for a new EpiCenter needs to be populated with dates appropriate for your installation. You can do this after you create your EpiCenter when you generate the schema as described in “Generating Schema” on page 214.

Optionally, you may choose Populate Date Dimension from the EpiCenter menu, and use the Date Dimension dialog box (Figure 13). Select the values for the beginning and ending year of the EpiMart and your calendar type and start day. Click Build to populate the EpiMeta database with a base list of dates.

Note: Quarters 4-4-5 represents the 13-week-per-quarter calendar in which the months in the quarter are defined as consisting of 4 weeks, 4 weeks, and 5 weeks.

Working with an Existing EpiCenter

For descriptions of these date fields, see “General Settings” on page 285 in and Appendix D, “Physical Type Values”.

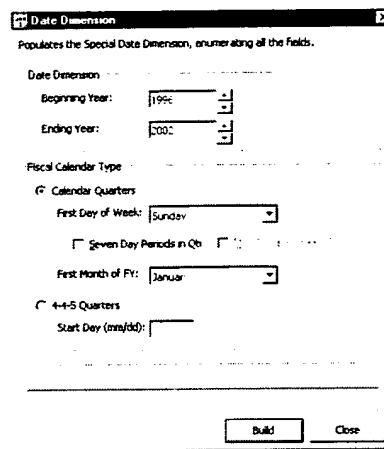


Figure 13: Date Dimension Dialog Box

For a discussion of how the Epiphany system treats time, see “The Uniform Treatment of Time” on page 31.

WORKING WITH AN EXISTING EPICENTER

If an EpiCenter has already been initialized, you may register it as follows:

- Step 1:** In the EpiCenter Enterprise Manager window, right-click the EpiCenters folder icon and select Register from the pop-up menu.
- Step 2:** In the Choose EpiCenter dialog box, select the EpiMeta for the EpiCenter.

The EpiCenter’s hierarchical tree is displayed in the EpiCenter Manager Enterprise window. You may modify this EpiCenter as appropriate for your site by following the instructions given in this chapter for those items you wish to change.

You can unregister an EpiCenter by right-clicking the EpiCenters folder icon (see Figure 10, on page 89) and selecting Unregister from the pop-up menu. The EpiCenter's hierarchical tree is removed from the window.

THE EPICENTER MANAGER WINDOW

Similar to other Windows applications, you may invoke EpiCenter Manager commands by using the main menu, toolbar icons, or by right-clicking icons in the window and selecting commands from a pop-up menu. Before you start configuring your EpiCenter, familiarize yourself with the main menu commands and the toolbar icons. The main menu commands are organized by the main EpiCenter Manager functions: Server, EpiCenter, Constellation, Extraction, Interface, Security, and Help. These menus are contextual and their commands are accessible when you are working within their function. For example, the Constellation commands are available when you working with a Constellation, but may be unavailable in other contexts.

EpiCenter toolbar icons provide access to special functions such as truncating tables, or displaying dialog boxes you use on a regular basis, such as the new Job dialog box and dialog boxes for defining facts, dimensions, external tables, and users and groups. A toolbar icon must be activated (highlighted) for you to use it. To activate a toolbar icon, select a related icon in the directory tree.

WORKING WITH EPICENTER MANAGER

The EpiCenter Manager window is organized in a hierarchical tree structure in which you right-click a plus or minus sign to expand or collapse a directory, as in Windows Explorer. In general, you will work down the tree structure in the window, right-clicking folders or icons to display a contextual pop-up menu, or you may prefer to use the equivalent main menu or toolbar commands. The instructions in this chapter use both ways interchangeably.

Working with EpiCenter Manager

Double-clicking an empty folder icon opens a dialog box in which you can define a new item, such as a new constellation. If this item has already been created, double-clicking expands or collapses the directory tree. Double-clicking an existing icon, such as the Default Job icon, opens its dialog box.

The following applies to all of your work with EpiCenter Manager:

- Descriptions in dialog boxes

The Epiphany system does not utilize the descriptions in the Description text area of the EpiCenter Manager dialog boxes. This description is for your documentation purposes only.

- Updating and refreshing

You may click the Update button in a dialog box to change the EpiMeta description of an existing row of data in the database “on the fly.”

If multiple users are running EpiCenter Manager, metadata changes made by one user are visible to others only after that user issues the Refresh command.

- Unregistering and deleting items

You can unregister servers and EpiCenters by right-clicking their icons and selecting Unregister from the pop-up menu. You can delete an item in the EpiCenter Enterprise Manager window, such as a base dimension table or constellation, by right-clicking its icon and selecting Delete, or by selecting it and pressing the Delete key.

- Duplication

The duplication feature allows entire objects to be copied or duplicated. (It is available by right-click in the main-menu, or by pressing Ctrl+C.) You are prompted to enter a name for the new object, which results in a complete “deep” (including all children) copy. This feature enables you to clone a base dimension table, fact table, extractor, and so forth.

EPICENTER MANAGER REPORT

The EpiCenter Report provides a record of all or part of your EpiCenter Manager configuration in an HTML file that displays in your browser window. (Figure 14 shows a sample report.) You can generate this report whenever you need a snapshot of your current EpiCenter (and use your Web browser to open saved reports for viewing). You may want to generate this report while you are developing your EpiCenter in order to record your progress.

To generate this report, either select one of your EpiCenters in the directory tree and choose Report from the EpiCenter menu, or right-click the EpiCenter folder and select Report from the pop-up menu. The Report dialog box (Figure 15) is displayed.

In the Report tab, enter the Report File name preceded by the directory location (If you do not enter this now, you will be requested to do so later.) If the Report File name is the same as another file, that file is overwritten.

Select all or a subset of the report topics. These topics reflect the major folders of the EpiCenter: Extraction, Interface, Security, Ticksheets, Schema, and Constellations. The Schema represents higher-level metadata that all constellations in the EpiCenter share.

You can display specific sub-topics; for example, selecting only Ticksheets results in a report that shows the attributes, measures usage, and filters for the EpiCenter. Optionally, you could select measure usage only as the report topic.

By default, the report is displayed after it is generated, and topics within the report are hyperlinked.

Click the Results tab to view the file location and status of the report's generation. The time that elapsed for a successful build is also given.

EpiCenter Manager Report

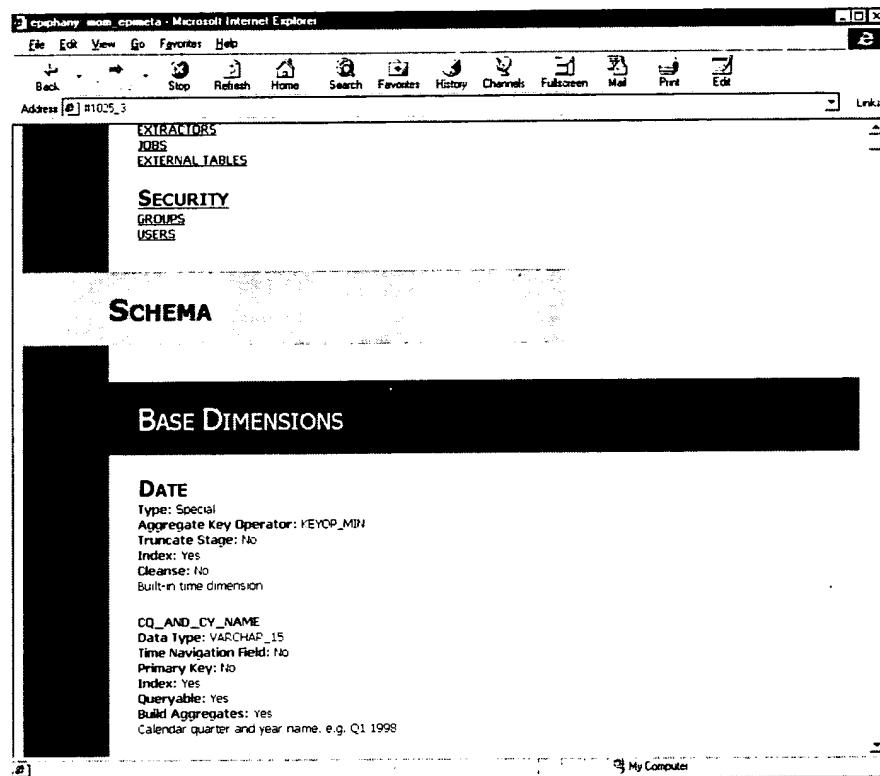


Figure 14: EpiCenter Manager Report

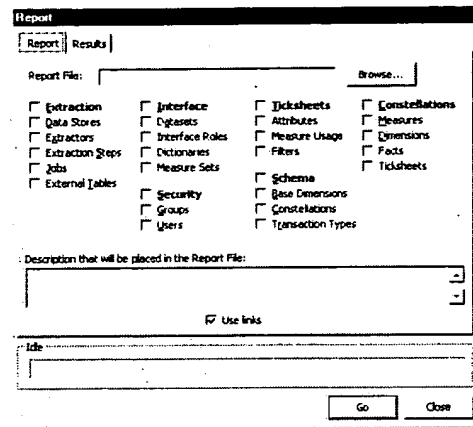


Figure 15: Report Dialog Box: Report Tab

SETTING UP AN EPICENTER

The purpose of setting up an EpiCenter is to define your site's metadata, or schema. After defining it, you will use the Generate Schema command in the EpiCenter menu to write these definitions to the EpiMart database. The EpiMeta database points to your EpiMart (where your extracted data is stored).

The instructions in the rest of this chapter describe how to set up a single EpiCenter with one constellation. Some EpiCenters, however, will have more than one constellation. To create additional ones, simply repeat the procedure you used to create your first constellation.

Configuration

The Configuration dialog box provides a record of the current configuration of your EpiCenter. It shows general configuration settings, lists transaction types, defines how measure units are displayed, shows the labels for display option names as they appear on ticksheets, and describes the ticksheet types. (Choose Configuration from the EpiCenter menu to view these settings.)

Base Dimension Tables

Other than entering your mail password in the General tab, the information in this dialog box should need little, if any modification. (Most of these settings can be set using other EpiCenter Manager dialog boxes, or are entered automatically as you configure the EpiCenter.) See Appendix B, “EpiCenter Configuration.” for more information.

Base Dimension Tables

The base dimension tables are physical dimension tables that can be used multiple times within a constellation or across constellations (depending on the dimension roles associated with a fact table). A base dimension table consists of dimension columns. These are columns in a physical base dimension table that hold attributes extracted from the source system.

All EpiCenters are created with two default base dimension tables: Date and Transtype (transaction type); therefore, the date and transtype dimensions occur in every fact table. (Remember that conceptually the star schema has the fact table in the middle and the dimension tables radiate outward as points of the star.) The use of a single date dimension throughout the system ensures a consistent treatment of time.

Transaction type dimensions are necessary because the EpiCenter must be able to distinguish among different rows in a single fact table, such as shipping transactions versus bookings, and bookings versus booked returns. For instance, an Order fact table might contain both a BOOK and SHIP transaction type, differentiated by the value of the column *transtype_key*. The *transtype_key* points to the TransType base dimension table. The transtype may be viewed as a “slice” of the fact table; semantically, the transtype slice acts as a separate fact table.

You will use the Base Dimension dialog box to—

- define the EpiCenter’s base dimension tables, such as Product, CostCenter, Account, Subaccount, or Project.
- configure the dimension aggregates used for aggregate building.

Base Dimension Tables

A typical star schema has multiple dimensions associated with a fact table. For example, an Order fact table of 20 million rows may have the dimensions Customer (1000 rows), Product (3000 rows), and Date (365 rows). In theory, you could construct a star schema that has only one dimension; that is, a supra-dimension that contains all three dimensions. The resulting dimension table, however, would need one row for every possible combination of dimensionality in the fact table.

Why not build a single dimension table in this case? The Customer, Product, and Date data is independent of each other. Three smaller dimension tables with 1000, 3000, and 365 rows individually combined with the fact table can represent the same information as the number of distinct combinations that actually occur in the data: 5,000,000. The only trade-off (other than that 5,000,000 rows in a dimension table exceeds the system limit) is that each dimension has an associated dimension role key in the fact table that is a 4-byte integer, and these bytes accumulate. In this extreme example, the extra dimension role keys are worth the extra space.

Sometimes combining two small, independent dimensions may make sense. For example, SalesPerson has ten distinct values in the Order fact table, and Promotion has 20 distinct values. Even if these two dimensions are completely independent, building a composite dimension with several hundred rows, which is very small, can save a key in the fact table. These are considerations to keep in mind when defining your base dimension tables.

To define a new base dimension table:

Right-click the Base Dimensions folder and select New Base Dimension from the pop-up menu. The Base Dimension dialog box (Figure 16) is displayed.

Base Dimension Tables

This dialog box has three tabs: General, Column Sets, and Built Aggregates.

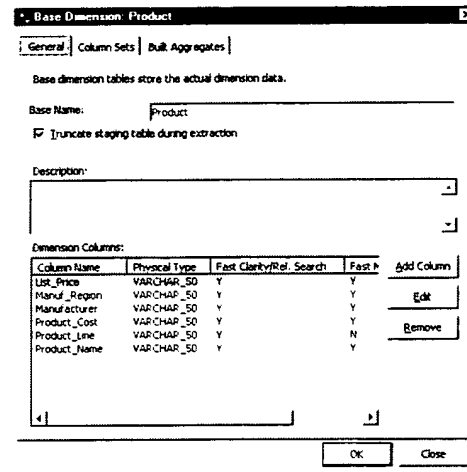


Figure 16: Base Dimension Dialog Box: General Tab

You can use the General tab to define the base dimension table:

- Step 1:** Enter the name of the base dimension table, such as *Product*. This is the name of the physical table in the database; underscore zero (_0) will be added to this name to form the base table. The maximum length is 20 characters.
- Step 2:** By default, the data in the base dimension staging tables is deleted (truncated) after extraction.

In most cases, you will want to truncate staging tables. You may, however, want to de-select the *Truncate staging table during extraction* option when an execute-only SQL statement truncates the staging tables based on criteria you specify for the extraction process. For example, this SQL could delete all records whose foreign keys matched, or all records over seven-days-old.

Alternatively, you may use a single dialog box to define all tables within a constellation that will be truncated prior to extraction (see “Truncating Tables” on page 198).

- Step 3:** Click Add Column to enter one of the table's column names.
- Step 4:** In the Dimension Column dialog box, enter a name and a description for the column.
- Step 5:** Selecting the *Indexed for fast Clarity and Relevance search* option includes this dimension column with other selected dimension columns in a inverse (non-unique) index for fast lookup. De-selecting this option for selected dimension columns saves space, but increases search time.
- Step 6:** *Can be used in Momentum filter* must be selected in order for this column to be moved into the Momentum tables.

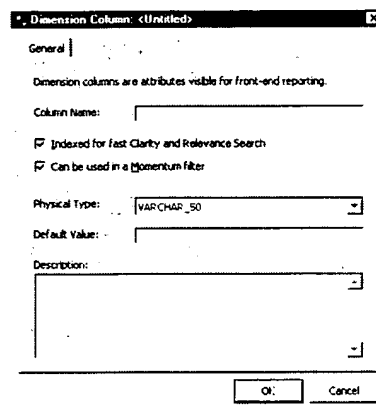


Figure 17: Dimension Column Dialog Box

- Step 7:** Select the physical type from the drop-down list. The physical type you select is replaced by its associated database type. The translation of physical type to database type depends on your RDBMS platform. See Appendix D, “Physical Type Values” for descriptions of these physical types.

Defining Dimension Aggregates

Step 8: Enter a default value for the column.

By default, each dimension column assumes the literal value UNKNOWN for each of its textual attributes. Leave this field blank to accept the default. To override this value so that end-user queries will return a different value, enter a literal string (without quotes). Your string will be used instead of UNKNOWN. See “The UNKNOWN Dimension Row” on page 64.

DEFINING DIMENSION AGGREGATES

A *dimension aggregate* represents a dimension table in which one or more of the columns have been removed, and the rows have been collapsed onto one another to remove duplicates. (In the example given in “Aggregate Building” on page 55, if Customer Name is removed from the base Customer table, only two distinct Regions are found in the table, so only two rows are needed in the Region aggregate table.)

You can use the Columns Sets tab of the Base Dimension dialog box (Figure 18, on page 104) to define which aggregates will be built the next time that the Aggbuilder program (**agg.exe**) is executed.

Column sets are subsets of the full list of dimension columns for the associated dimension base table. The base dimension column sets determine which base dimension aggregates the system builds. The number of rows in the dimension aggregate equals the number of distinct column set values. For aggregates, each column set represents a *potential* aggregate to be built. The Aggbuilder program does not build all defined column sets, it builds only those columns sets that are included in fact aggregates (see “Aggregation and Aggregate Grouping” on page 115).

Often one of the columns “drives” the set because sets usually represent a natural hierarchy in the data. The other columns are higher level roll-ups.

Defining Dimension Aggregates

When defining dimension column sets, keep the following in mind:

- Do not place two dimension columns with high-cardinality (many values), and whose data is independent of each other in the same set. For example, City and CustomerType both have thousands of distinct values and are unrelated to each other.
- The size of the dimension aggregates greatly influences the size of the fact aggregates.

To define dimension aggregates, follow these steps:

Note: The data you enter in the Column Sets tab is used by the Aggregate Building process the next time **agg.exe** is run. Your changes do *not* affect the current EpiMart tables.

Step 1: Enter the name of the column set in the text box, and click New.

This name is a logical identifier only, and is not used during the building of the Actual table in the EpiMart. Instead, EpiCenter Manager will append unique numbers after the base dimension table name.

Step 2: Select *aggregate* as the Set Type from the drop-down list. (Mini-dimensions are used by the Momentum application and are discussed in “Using Momentum” on page 160.)

Step 3: Select or de-select *Include in default aggregate groups*.

Selecting *Include in default* for this column set affects the behavior of the default aggregate group that exists in each constellation (see “The Default Aggregate Group” on page 117).

Step 4: Add column names for this set by clicking New. Select columns from the list of columns for this base dimension table.

Defining Dimension Aggregates

Base Dimension: Product

General | **Column Sets** | Bulk Aggregates

Column sets define which aggregates or non-dimensions will be built.

Set Name:

Set Type: ☒ Include in default aggregate groups

Set Name	Set Type	Default	New
----------	----------	---------	-----

Columns in the Set:

Column Name	Group By
-------------	----------

OK Close

Figure 18: Base Dimension Dialog Box: Column Sets Tab

Step 5: Select or de-select Group By.

When including a column in a set, you may specify whether that column is to be used in the GROUP BY clause of the SQL statement that generates the aggregate. Normally, you should select all Group By flags. If a column's value is determined by another column already flagged as Group By, then the former column can be left out of the GROUP BY clause without affecting the result set.

For example, consider a dimension called Product which has, among other things, both a category code and a category name. If each code has a unique associated name, then the code and name should be included in the same set because any aggregation on one is an aggregation on the other. Because the two columns change together, there is no point to group by *category_number* and *category_name* because either group by produces the same results. One of these columns can be left out of the group-by list.

Add additional (multiple) column sets by repeating the above steps. See the next section for more information.

Multiple Dimension Column Sets

Multiple Aggregate set types are used when a base dimension table with many columns can be aggregated in different ways. Generally, the fewer columns included within a column set, the smaller the resulting dimension aggregate, and the faster queries that use fact aggregates joined to that dimension aggregate will respond.

For instance, if a customer dimension has several fields related to customer geography (City, State, Zip, and so forth) and several other fields related to demographics (age, marital status, years of education), then EpiCenter Manager can choose two or more dimension column sets (which can overlap) for maximum coverage of queries. At query time, Epiphany's aggregate navigation mechanism will choose the smallest aggregate that satisfies the user's request. Thus it is beneficial at query time to have built many small, highly specialized aggregates. The penalty for building many dimension aggregates is that it takes longer to build aggregates, and requires more disk space.

Note: A small change in the number of dimension aggregates can cause a large change in the number of fact aggregates. The Aggbuilder program builds only those column sets included in fact aggregates (see "Aggregation and Aggregate Grouping" on page 115).

Dimension Aggregate Browsing

While configuring your Epiphany system, you may wish to browse the list of aggregates that Aggbuilder has built (according to the instructions in the Aggregate Group dialog box described in "Aggregation and Aggregate Grouping" on page 115). You can use the Built Aggregates tab of the Base Dimension dialog box to browse the aggregates for either the A or B set of tables.

Setting Up a Constellation

EpiCenter Manager uses a mirroring structure in which all EpiMart tables are in the database twice, once for each of the _A and _B tables. Only one set of tables, however, is active at a given time. See “Mirroring: A and B Tables” on page 83 for more information.

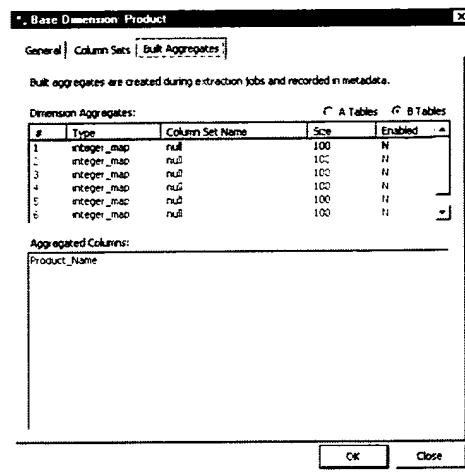


Figure 19: Base Dimension Dialog Box: Built Aggregates Tab

SETTING UP A CONSTELLATION

Within an EpiCenter, constellations serve to group fact tables that have similar dimensions. Instructions are given here for setting up a single constellation for Clarity and Relevance. To set up another constellation, repeat the entire series of steps in this section.

Note: For instructions on setting up a Momentum constellation, see “Using Momentum” on page 160.

To set up a constellation for Clarity and Relevance:

Step 1: Choose New Constellation from the EpiCenter menu.

Step 2: Enter the new constellation name and any description.

The New Constellation icon appears in the Constellation folder. It has these sub-folders Facts, Dimension Roles, Aggregates, Measures, and Ticksheets. Although you can usually proceed down the tree as you define your EpiCenter, you need to define dimensions and facts before aggregates.

Defining Dimension Roles

A base dimension table may have a superset of data that can be used for different purposes. For example, a Customer base dimension table may contain Bill-to and Ship-to data. Dimension roles enable base dimension tables to be used more than once in a constellation.

A dimension role indicates the single usage (such as Bill-to data only) of a base dimension table by all of the fact tables within a constellation. A row in a fact table may have several foreign keys that point to the same base dimension table, but the role usage may differ (for example, some relate to Bill-to and others to Ship-to data). The dimension role maps the fact foreign keys to the correct base dimension table.

To define roles for this dimension table:

- Step 1:** Right-click the Dimension Role folder and select New Dimension Role. The Dimension Role dialog box is displayed.
- Step 2:** Select the base dimension name from the drop-down list.
- Step 3:** Enter the Dimension Role Name and a description, and click OK. The dimension role icon is added to the tree.
- Step 4:** Repeat the above steps for any other base dimension tables that have multiple roles.

Every dimension role must have a parent base dimension table. If there is no base dimension for this role, click New in the Dimension Role dialog box. You can use the Base Dimension dialog box that displays to create a new base dimension.

Degenerate Dimensions

Degenerate Dimensions

Certain numbers in fact tables such as order, invoice, and bill of lading numbers have foreign keys with no corresponding dimension table. The only data of importance is the number itself, all of the other information of value has been extracted into other dimension tables. These numbers are called degenerate dimensions.

A degenerate dimension is a textual field stored directly in the fact table. It is part of a constellation, and all facts in the constellation inherit it. If you use degenerate dimensions, you need to inform the system of their existence. To define a degenerate dimension:

Note: Choose New Degenerate Dimension from the Constellation menu. Enter the degenerate dimension name and a description in the Degenerate Dimension dialog box.

Defining Fact Tables

A fact table is a physical table that holds numeric data in its columns. It also represents the intersection of a series of dimension keys. A fact table consists of dimension role foreign keys, degenerate dimension keys, and fact columns.

As mentioned, the transtype (transaction type) dimension occurs in every fact table. The transtype is a “slice” of the fact table that functions as a separate fact table. You can configure one fact table with multiple transtype base dimensions, or configure multiple fact tables. The advantage of having multiple transtypes is that measures that require two transaction types, such as BOOK and BOOK_RETURN need to issue only one query. The advantages of multiple fact tables are that queries that need only one transtype have fewer rows to navigate, and that aggregate tables are smaller.

To define a fact table:

Step 1: Right-click the Facts folder icon, and select New Fact. The Fact Table dialog box (Figure 20) is displayed. This dialog box has tabs labeled General, Custom Index, and Built Aggregates.

Defining Fact Tables

If Momentum is installed, there is also a tab named Momentum, which applies only to fact tables in Momentum Adjacent constellations. You will use this tab when you set up a Momentum constellation as described in “Using Momentum” on page 160.

- Step 2:** In the General tab, enter the fact table’s name and text that describes this fact table. The maximum length is 20 characters.
- Step 3:** By default, the data in the fact staging tables is deleted (truncated) after extraction. In most cases, you will accept the default.

Fact Table: Bought

General | Custom Indexes | Momentum | Built Aggregates

Fact tables store the transactions in Epimart. A transaction represents the intersection of dimension values along with numeric values describing that event.

Fact Name: Bought

Description:

☒ Truncate staging table during extraction

Fact Columns:

Column Name	Physical Type	Queryable
Item_Count	FACTQTY	1

Add Column Remove Edit

OK Close

Figure 20: Fact Table Dialog Box: General Tab

Defining Fact Columns

Defining Fact Columns

Fact columns contain the actual customer numeric data; such as, *net_price* or *number_units*. You will use the General tab of the Fact Table dialog box to define a fact column:

Step 1: Click Add Column. The Fact Column dialog box (Figure 21) is displayed.

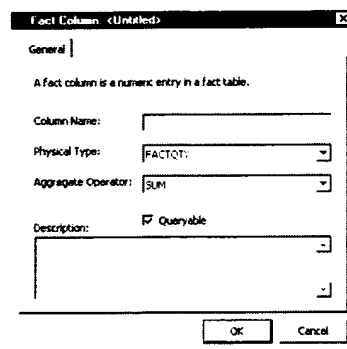


Figure 21: Fact Column Dialog Box

Step 2: Enter the column name and select the physical type from the drop-down list. The physical type you select is replaced by its associated database type. The translation of physical type to database type depends on your RDBMS platform. See Appendix D, "Physical Type Values" for descriptions of these physical types.

Note: For this release, the FACTMONEY, FACTQTY, and EPIINT physical types are supported for fact columns.

Step 3: Select the aggregate operator from the drop-down list. Aggregate operators determine how the Aggbuilder program calculates the facts. (For this release, SUM is the only option.)

Step 4: The Queryable option is selected by default. If this option is not selected, the column will not be moved into the Momentum tables.

Step 5: Enter a description, and click OK to return to the Fact Table dialog box.
The new fact column appears in the listbox.

Step 6: Repeat the above steps to define another column.

To remove a column name, select it and click Remove.

Custom Fact Indexes

In general, aggregates satisfy high-level queries while indexes satisfy highly selective queries. Query drill-downs on a ticksheet transition from broad to selective. In terms of aggregates and indexing, the sequence of a drill-down transition from top to bottom might be—

Step 1: A small aggregate (for example, Business Unit equals Copiers).

Step 2: A larger aggregate (month equals January 1999).

Step 3: An index on a large aggregate (Customer Region equals West).

Note: Aggregate tables are indexed the same as their base tables on those columns that still exist in the aggregate. Aggbuilder does not build the same index twice simply because of missing columns.

Step 4: An index on a base dimension table (drill down by Customer Name equals John Doe).

A custom fact index is the metadata definition of indexes to build on fact tables in EpiMart. It is an ordered list of dimension roles. You can use EpiCenter Manager to configure custom indexes to dramatically improve query performance for selective queries. To actually build custom indexes, however, you must use the semantic template Custom Fact Index, which is described in Appendix F, “Semantic Types.”

By default, each fact table has a single clustered index with the leading term of *date_key* which allows fast filtering based on time. (The leading term of an index is the most important term; for example, a phone book is indexed by last name; you cannot locate someone by first name only.)

Custom Fact Indexes

If end users typically apply selective filters to other dimension roles such as customer or product, then configuring custom indexes with these leading terms will improve system performance. (See “Custom Fact Indexing” on page 62 for a more in-depth discussion of this topic.)

You can use the Custom Indexes tab of the Fact Table dialog box (Figure 22) to create a new custom index:

Step 1: In the Indexed Columns panel, click New.

Step 2: Select one or more dimension roles from the Choose dialog box to add to this index. The first dimension role will be the leading term.

To delete the index, select it and click Remove.

You can use the Up and Down buttons in the Dimension Roles panel of the dialog box to change the order of the dimension roles in the index.

To modify dimension roles for an index: In the Dimension Roles panel, select the index and click Add Dim to add additional roles.

To remove dimension roles from an index, select the index in the upper panel. In the lower panel, select a dimension role and click Remove.

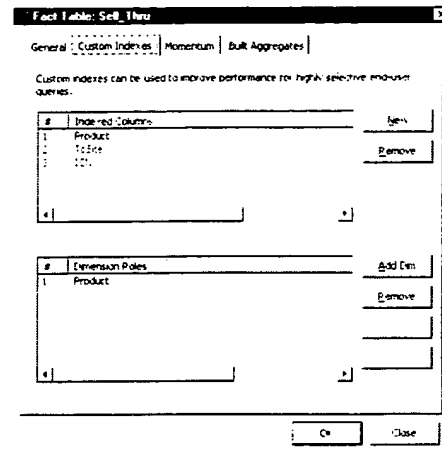


Figure 22: Fact Table Dialog Box: Custom Indexes Tab

Fact Aggregate Browsing

While configuring your Epiphany system, you may wish to browse the list of built fact aggregates to determine which tables are actually available for the query machinery. (See “Aggregation and Aggregate Grouping” on page 115 for more information.)

You can use the Built Aggregates tab of the Fact Table dialog box (Figure 23) for this purpose. As with dimension aggregate browsing, you can browse either the A or B set of tables, as appropriate. Follow these steps:

Step 1: In the upper panel of the tab, select a dimension role from the Role listing and then select a filter from the drop-down list. (Make one selection per dimension role.)

Filtering specifies that a dimension role—

- should not be included in the aggregate. (Not Included)
- should be included as the base table. (Base)

Fact Aggregate Browsing

- should be included as a specific dimension aggregate (corresponding to a dimension column set). (No Filter Applied)

Other filters are user definable in the Column Sets tab of the Base Dimension dialog box.

Step 2: Fact aggregates appear in the Fact Aggregates listbox.

Each fact aggregate has an associated number. Although this numbering is arbitrary, it may be helpful when debugging query logs since the logs use these numbers. The size and compression ratio (number of rows in this aggregate divided by the number of rows in the base fact table) are also given. You can click the Size or Compression headings to sort aggregates by size. This sorting shows which aggregates are most valuable (smaller compression ratios mean greater aggregation).

Clicking a row in this listbox displays the following in the Aggregate Contents panel of the tab: the dimension roles that are included for this fact aggregate, and the dimension aggregates (or the base table) that the fact aggregate joins to.

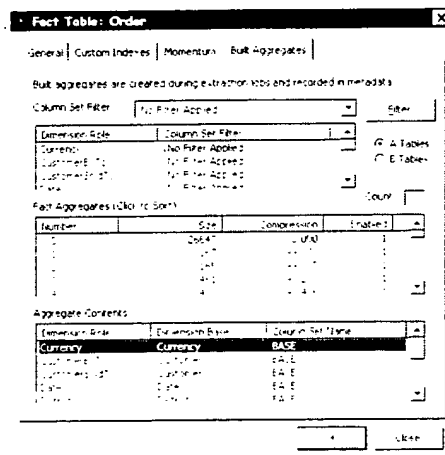


Figure 23: Fact Table Dialog Box: Built Aggregates Tab

Aggregation and Aggregate Grouping

Aggregates are facts that the system pre-calculates for a group of dimensions. (These dimensional groupings are determined by the groups you set up for a base dimension table.) An *aggregate group* is a set of instructions (a metadata definition) that tells Aggbuilder which aggregates to build on one or more fact tables. The fact aggregates that are actually built are determined by a combinatorial expansion based on the dimension role possibilities.

Unlike dimension aggregates, fact aggregates are not fully enumerated through the use of EpiCenter Manager. The reason for this is simple: while base dimensions typically have only a few aggregates, facts usually have many.

EpiCenter Manager simplifies aggregation by providing aggregate groups that you define using the Aggregate Group dialog box (Figure 24, on page 117 shows this dialog box for the Default Aggregate Group). For each aggregate group, you will specify the facts in the constellation that share it. (An aggregate group applies to a single constellation.) If two or more facts share an aggregate group, then the Aggbuilder program will follow the same set of instructions for each of the facts.

In general, when building aggregates, aim for a combination of broad fact aggregates (some compression/high query coverage) and narrow fact aggregates (high compression/low coverage).

To open an existing Aggregate Group dialog box, double-click it. (To open a new Aggregate Group dialog box, expand the Constellation tree and right-click the Aggregates folder. Select New Aggregate from the pop-up menu.)

You can use the General tab of the Aggregate Group dialog box to define an aggregate group:

- Step 1:** Enter an aggregate group name (and description).
- Step 2:** Select Enabled for Aggbuilder for the Aggbuilder program to use this grouping.
- Step 3:** Select the facts that share this aggregate. Click Add Fact and select facts from the Choose dialog box.

Aggregation and Aggregate Grouping

Dimension roles in the aggregate group determine how fact aggregates are produced. The choices for inclusion or exclusion of a dimension role in an aggregate group are—

- **Exclude the Dimension Role**
Leave the dimension role out of the fact aggregate completely. If a dimension role is not included in the aggregate group, then *all* aggregates that result from this aggregate group do not include that dimension role.
- **All Columns in the Base Dimension**
Perform no aggregation on that dimension role.
- **Column Set**
Perform aggregation on the dimension columns that comprise the Column Set. When dimension aggregates are built during the aggregation process, each column set for a base dimension will become a dimension aggregate.

You can use the Definition tab of the Aggregate Group dialog box to add dimension roles for the aggregate group and to specify the ways to include them:

- Step 1:** In the upper panel, Click the Add Dim button.
- Step 2:** Select the dimension roles to be considered for these aggregates from the Choose dialog box.
- Step 3:** Select the ways to include the selected dimension role in fact aggregates by selecting a dimension role in the upper panel and clicking Add Set.
- Step 4:** Select the column sets for this dimension role.

The total number of Aggregates is shown on the tab.

The Default Aggregate Group

Each constellation has a default aggregate group that has special semantics to make the configuration of aggregates simpler. You cannot delete this group, although you may disable it.

The date dimension is automatically included in this group, since many end-user queries involve time. Date is included as an Exclude Dimension Role and Column Set. The Column Set is determined by which column sets of the Date Dimension dialog box are set to Default.

When a new dimension role is added to the constellation, the default aggregate group is automatically modified to include that dimension role. All Column Sets declared as Default for that base dimension table are included in the set of column set possibilities for the dimension role.

In all other respects, the Default Aggregate Group is identical to any other aggregate group.

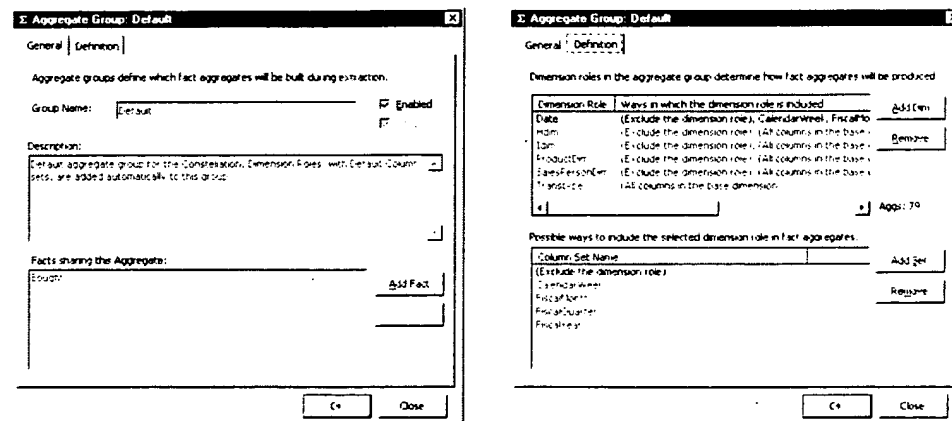


Figure 24: Default Aggregate Group Dialog Box

Measures

MEASURES

A *measure* is a business calculation that is the result of an arithmetic combination of fact columns. Each constellation has associated measures, which are organized alphabetically by name in the EpiCenter Manager's Measures folder.

When a user selects measure column *elements* in ticksheet columns, such as Average Sales Price, BookedOrders, and Gross, the calculations that apply to this combination of those elements depend on the measure to which they are mapped (for example, ASPBookedOrdersGross). These calculations are the values returned in the cells of Clarity and Relevance reports.

This section describes how to define measures and how to map them to column elements. *Measure mapping* is the means by which the Epiphany system knows which measure to apply when a user selects measure column elements from ticksheet columns. (Measure mapping does not apply to Relevance Influence ticksheets.)

Defining a Measure

To define measures for the constellation, use the Measure dialog box (Figure 25):

Step 1: Right-click the Measures folder and select New Measure from the pop-up menu. The Measure Builder dialog box (Figure 25) is displayed.

Note: Changes you enter in this dialog box are saved when you click OK, or simply close the dialog box. There is no Cancel button.

Defining a Measure

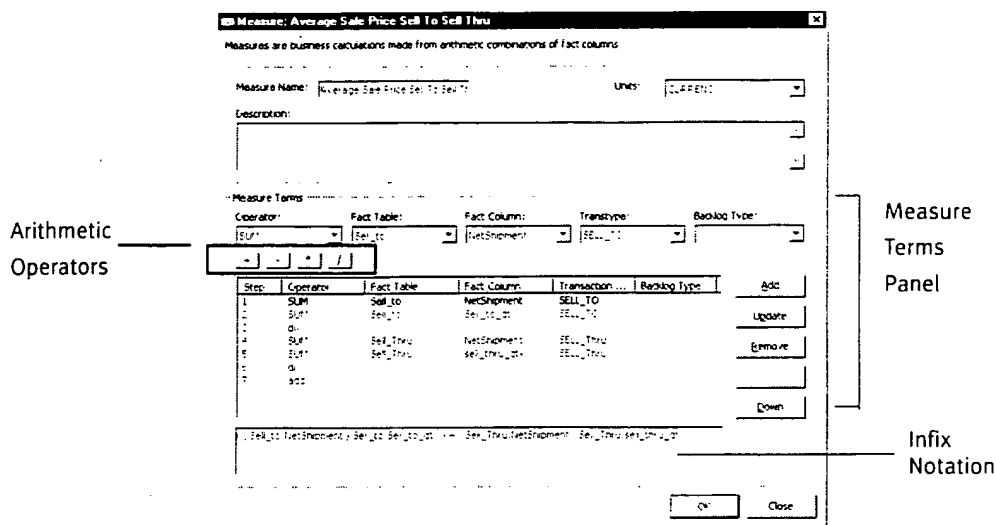


Figure 25: Measure Dialog Box

- Step 2:** Enter a name for the new measure. Typically, measure names are a combination of the column element names; for example, the measure ASPBookedOrdersGross indicates that the user selected Average Sales Price, Booked Orders, and Gross.
- Step 3:** Enter text that defines this measure for your reference.
- Step 4:** Select the unit of measurement from the Units drop-down list.
You can define unit options via the Measure Units tab of the EpiCenter Manager's Configuration dialog box (see "Measure Units" on page 289).

Defining a Measure

Example units of measurement are—

- Currency (for money units)
- Per cent (for percentages)
- Units (for the count of an item)

Measure Terms

You can use the Measure Terms panel of the Measure dialog box (see Figure 25) to define the steps that determine how the measure is calculated. A *measure term* is one component of an arithmetic expression that makes up a measure. A measure term refers to the aggregation of a single fact column in a fact table, such as *SUM(Order.net_price)* with a particular transaction type. Each term applies to a specific fact table, fact table column, transaction type, or backlog type (if applicable).

One measure term is combined with other measure terms to create a composite measure. The Epiphany system converts these steps to appropriate SQL SELECT statements. You will use Reverse Polish Notation to construct measure definitions. See “Reverse Polish Notation” on page 122 for more information.

To define a measure term step:

Step 1: Select one of the SQL operators from the drop-down list: SUM, COUNT, COUNT DISTINCT (or the negative values of these operators, such as -SUM or -COUNT DISTINCT).

Step 2: Select the fact table where the data resides, the fact column, transaction type, and backlog type.

The fact column is one of the columns of the fact table. All facts have an associated transaction type. Transaction types function as “slices” of a fact table; for example, all facts that are booked returns can be viewed as one slice of the fact table. Semantically, this slice functions as a separate fact table. For a description of the kinds of transaction types, see Appendix B, “EpiCenter Configuration.”

Defining a Measure

The backlog types are BEGIN or END; or leave blank if the backlog type does not apply. You can use a backlog type when a measure term (a single line of the measure definition) should exhibit accumulating behavior. Normally, when running a report of Sales by Month, for example, the report shows only the sum of transactions that occurs in each month of the report. When a backlog type is used, however, the columns of the report show accumulated values from previous months, in addition to the current month. You can use BEGIN to show the accumulated value at the beginning of each period, and END to show the ending accumulated value.

For example, assume that report of sales by month transactions shows \$10 for June, \$20 for July, and \$40 for August. A report of the beginning backlog shows the accumulated value at the beginning of each month:

June	July	August	September
\$0	\$10	\$30	\$70

A report of the ending backlog shows the accumulated value at the end of the month:

June	July	August
\$10	\$30	\$70

- Step 3:** Click Add to add this as the first step in the Measure Terms panel.
- Step 4:** If appropriate, add another measure term by repeating Steps 1 through 3.
- Step 5:** Click the appropriate arithmetic operator to be applied to the measure terms: add (+), subtract (-), multiply (x), or divide (/). See “Reverse Polish Notation” on page 122 for examples.

Reverse Polish Notation

The bottom panel of the Measures dialog box (see Figure 25) displays a read-only infix display, or representation of the measure calculation, which translates the Reverse Polish Notation calculations to standard calculations. If the measure calculation does not currently make sense (for example, Term, Add, Add), the notation in the lower panel displays in red with one or more missing or extra term tags. You can save the measure, but it will not work in the Application Server.

Important: The removal of fact or dimension columns can remove measure terms, which invalidates their calculations. In this case, check all of the measures to determine if there are any infix displays in red.

To edit an existing measure term, right-click its Measure folder and select Edit from the pop-up menu, which displays its Measure dialog box. Modify this as described above. Click Update to show the changes in the dialog box, and click OK to save them.

To delete a measure, right-click its folder and select Delete from the pop-up menu. You can also use this pop-up menu to duplicate or export the measure.

Reverse Polish Notation

The Measure Terms panel in the Measure Builder dialog box uses Reverse Polish Notation³ (RPN) to construct measure definitions. RPN operations are performed in a last-in, first-out (LIFO) basis. All of the values to be operated upon are placed in a stack. Then the top two are operated upon and the result of that operation is placed in the stack, replacing the previous two values. Then the next top two are operated on and the result placed in the stack, and so forth.

For example, using Reverse Polish Notation for this calculation:

$1 + (2 * 3) = 1, 2, 3, *, +$ in RPN

means that 1, 2, and 3 are placed in the stack. The last two items (3 and 2) are multiplied, and the resulting value 6 is placed in the stack. The stack now holds 6 and 1, which are added, and the result 7 is placed in the stack.

³ Reverse Polish Notation is named for its Polish inventor, Jan Lukasiewicz.

In contrast, applying RPN to the calculation:

$(1 + 2) * 3 = 1, 2, +, 3, *$ in RPN

means that 1 and 2 are placed in the stack. These items are added, and the result 3 replaces them. Next, the value 3 is placed in the stack (the stack now holds 3 and 3). These items are multiplied and the result is 9.

Example 1 below shows how RPN is used to define a measure definition (in the Measure Builder dialog box). The Average Sales Price (ASP) for Booked/Gross Orders equals the total number of dollars received, divided by the total number of units shipped. This is represented as the following SQL SELECT statement:

```
SUM (Order.net_price)
SUM (Order.number_units)
div
```

Example 1

This SELECT statement is calculated using RPN as follows. The sum of all the Order net prices is calculated and placed in the stack. Then the sum of all the number of units is calculated and placed in the stack. Next, the division operator is applied to the top two items in the stack. It takes the next item in the stack, the total net price received, and divides it by the total number of units shipped, and the result equals the ASP.

For the operators that apply to the aggregates, use the arithmetic operators: add, sub, mult, and div.

Example 2 shows a similar calculation. This time the ASP is calculated for booked net orders (booked orders minus returns). Here the sums of two Order net prices are added (the returned items are represented as a negative number) and placed in the stack. Then the sums of two Order number of units (the returned items are represented as a negative number) are added and placed in the stack. The remainder of the calculation is the same as for Example 1 above.

Ticksheet Types

```
SUM (Order.net_price)BOOK
SUM (Order.net_price)BOOK_RETURN
add
SUM (Order.number_units)BOOK
SUM (Order.number_units)BOOK_RETURN
add
div
```

Example 2

TICKSHEET TYPES

A ticksheet is Epiphany's user-interface form that allows authorized users to construct queries. It is called a *ticksheet* because users select, or tick, items on a page that they open in their Web browser.

A description of the Clarity, Relevance, and Momentum ticksheet types follows.

Clarity

- Table & Charts.

A Clarity ticksheet produces reports for selected attributes (the columns and rows of the report) with measure values inside the cells. Corresponding bar and pie charts are also provided.

Relevance

- Best & Worst

Relevance Best & Worst searches thousands of Clarity tables automatically and builds a list of the most interesting results within those tables.

- **Influence**
Relevance Influence builds models of a company's data to help identify the factors with the greatest impact on the business.
- **Lifecycles**
Relevance Lifecycles predicts performance of new business entities, such as products, customers, sales representatives, and so forth based on the historical performance of related entities.
- **Profiling**
Relevance Profiling shows a page of thumbnail charts that help you quickly understand your data.
- **Quarter Projections**
Relevance Quarter Projections answer the question, "Given where I am today, and given the history of previous quarters, how is this quarter likely to end? Will I meet my goals?"
- **Trends**
Relevance Trends fits a smooth curve to time-varying data, and forecasts one or more periods into the future.

Momentum

Momentum generates lists using demographic data for individuals or groups. You can use Momentum to define and generate lists of people or things that match your requirements. For example, Momentum can provide a list of all customers with outstanding invoices over 90 days, or all customers who have purchased one set of products but not another. You can also use Momentum lists as filters in Clarity and Relevance.

The Momentum ticksheet types are Individual and Household.

CONFIGURING A CLARITY OR RELEVANCE TICKSHEET

Note: The Ticksheet Types tab of the Configuration dialog box (Figure 82, on page 294) enables you to disable an entire application by choosing the application from the drop-down list and de-selecting Enabled. For example, you may want to test Clarity ticksheets, but have not yet configured any Relevance or Momentum ticksheets.

To start building a new Clarity or Relevance ticksheet, right-click the Ticksheets folder and select New Ticksheet from the pop-up menu. The appropriate Ticksheet dialog box for the ticksheet type is displayed. A ticksheet dialog box has six tabs: General, Attributes, Filters, Measure Mappings, Files, and Copy Ticksheet Items. (Relevance Influence ticksheets have the Measure Sets tab in place of Measure Mappings; additional instructions for configuring Influence are given in “Relevance Influence Ticksheets” on page 152).

First, configure a new Clarity or Relevance ticksheet (other than a Relevance Influence ticksheet). Instructions are given below for completing each tab in the Ticksheet dialog box.

Note: The configuration of Relevance and Momentum ticksheets differs in some aspects. These differences are addressed in “Configuring Relevance Ticksheets” on page 146 and “Using Momentum” on page 160.

General Tab

Step 1: You can use the General tab to enter the name of the ticksheet (no spaces are allowed). Any descriptive text is for your reference only.

Step 2: For the Ticksheet Type, select the kind of ticksheet you are building from the directory tree, such as the Best & Worst ticksheet in the Relevance folder. (Click a plus sign to expand the tree.) The kinds of ticksheets are described in Ticksheet Types above.

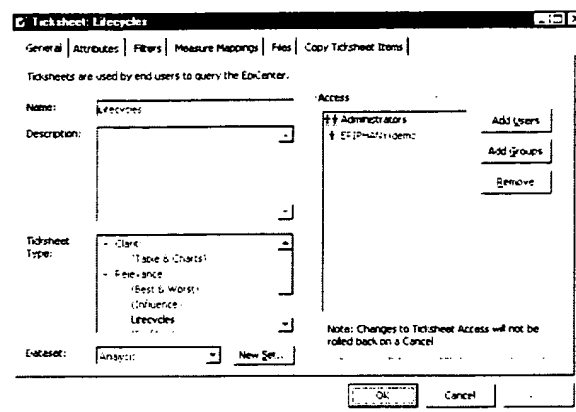


Figure 26: Ticksheet Dialog Box: General Tab

Assigning the Ticksheet to a Dataset

Datasets are subsets of the constellation's ticksheets listed in the left frame of Clarity, Momentum, and Relevance Web pages.

Step 1: Click New Set to create a new dataset. The Dataset dialog box (Figure 27) is displayed. Enter the dataset name and the label name (the name as it should appear on the ticksheet). Enter any help text that will be displayed to end users.

After data sets have been created, you can select them from the drop-down list. (See "Setting Up Datasets" on page 213 for more information.)

Step 2: In the General tab, enter the dataset name (the name used internally by the system) and the label name (the name that will appear in the left frame of the ticksheet under the Dataset heading as shown in Figure 28).

Step 3: Enter any help text that will be displayed to end users in the Help Text box.

General Tab

Step 4: The Tickshéets tab shows the ticksheets assigned to this dataset.

Note: Datasets are organized in the Shared Interface folder in the EpiCenter Manager directory tree. To rearrange the order in which the Datasets appear to end users, right-click a dataset in the Shared Interface folder, and select Up or Down from the pop-up menu.

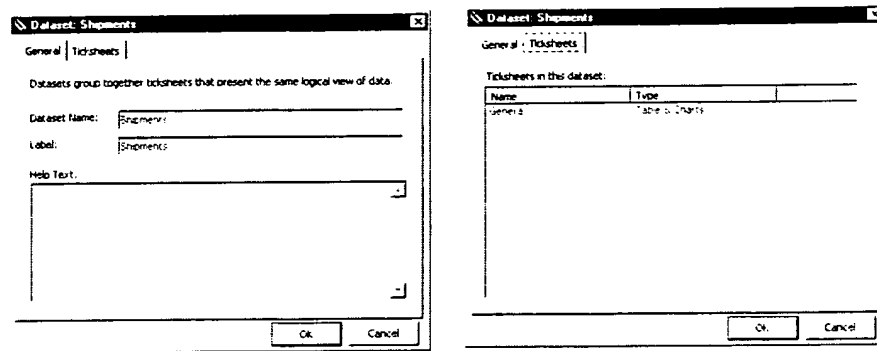


Figure 27: Dataset Dialog Box

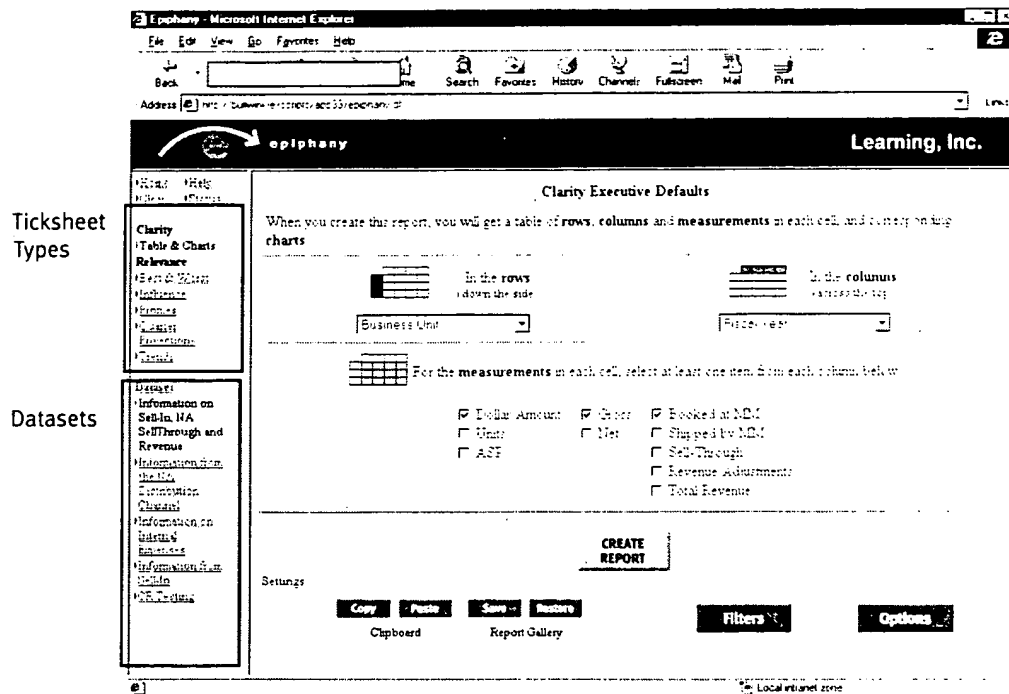


Figure 28: Ticksheet Types/Datasets in Ticksheet Menu

Assigning Access to Groups and Users

Before Epiphany users can open any ticksheet, they must be granted access to it. Users have the ability to see and use all ticksheets to which they have access, as well as to all ticksheets granted access to their groups.

New users and groups are set up using EpiCenter Manager's Security folder (see "Security" on page 200 for instructions). In the Access panel of the General tab of the Ticksheet dialog box (Figure 26), you will select which of those users and groups can view this ticksheet.

Attributes Tab

- To allow access to an individual user, click Add User and select the user from the Choose User dialog box.
- To allow access to a group, click Add Group and select the group from the Choose Groups dialog box.

Note: Because the Access information is owned by users and groups, and not ticksheets, these assignments take effect immediately. Changes are not rolled back when you click Cancel in the Ticksheet dialog box.

Attributes Tab

In Clarity, the attributes, which derive from the dimension tables, are arranged by columns and rows. For example, if you select products for the column and years for the row, the report will have columns for each of the designated products in the database and rows for each of the designated years. In Relevance ticksheets, you may select attributes in order to forecast trends, or to analyze highs and lows, best and worst cases, or graphs for various aspects of your business.

You can use the Attributes tab (Figure 29) to set up the attributes for the ticksheet.

Attributes Tab

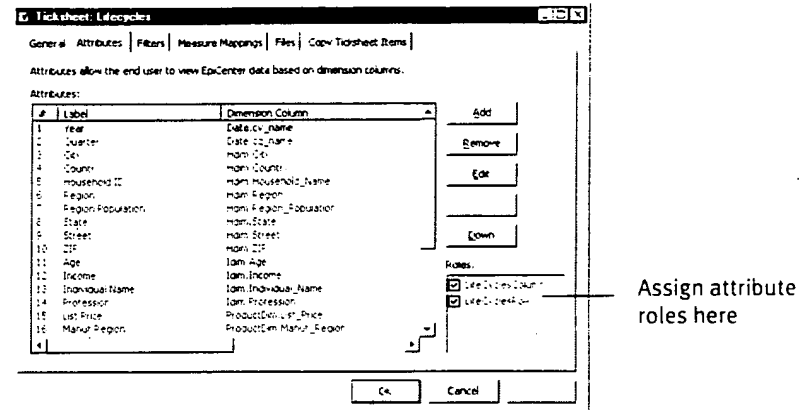


Figure 29: TickSheet Dialog Box: Attributes Tab

- Step 1:** Click the Add button, which displays the Attribute dialog box (Figure 30).
- Step 2:** Enter the attribute's Label (the name that appears on the ticksheet).
- Step 3:** Enter an abbreviation that the system may apply if the Label name is too long (for example, ASP for Average Sales Price).

Attributes Tab

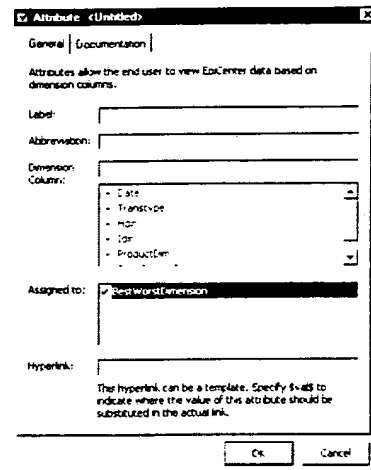


Figure 30: Attribute Dialog Box

- Step 4:** Select the dimension column for this attribute from the base dimension table listing. (Click a plus sign to expand the tree.)
- Step 5:** Assign attribute roles, such as use for both a column and row in a Clarity ticksheet, by selecting the role in the lower-right area of the dialog box (see Figure 29). (The ticksheet type determines these options; see “Ticksheet Types” on page 124.) You can also use the Roles panel of the Attributes tab to do this.
- Important:** Attributes can be defined only once per ticksheet. If you want a single attribute to appear in more than one place on a ticksheet, you must assign it multiple roles. (Clarity and some of the Relevance ticksheets have multiple roles.)
- Step 6:** The Hyperlink text box enables you to use the attribute as a link. For example, companies in a customer list could be linked to Web sites, such as <http://www.co-select.com>. You can use the string `val` to indicate where the attribute is to be substituted; for example: `www.val-select.com`.

Step 7: You can use the Documentation tab to enter a description for internal use only, or to make a glossary entry for this attribute.

Attributes and other ticksheet terms can be hyperlinked to a glossary page. When a user clicks a link, a glossary page displays which defines it. These glossary entries help users to understand your terminology. See “Setting Up a Glossary Entry” below.

Step 8: In the Attributes dialog box, click OK to add the attribute. It now appears in the list in the Attributes tab. Follow the same procedure to create (or modify) additional attributes.

Step 9: Select an attribute and click Up and Down in the Attributes tab to rearrange the order in which the attribute appears on the ticksheet.

Step 10: If you have not already done so, assign a selected attribute the application roles (such as ClarityColumn and ClarityRow) that are appropriate for the ticksheet type.

To modify an attribute, select it from the list in the Attributes tab and click Edit, which opens its Attribute dialog box.

Filters Tab

Setting Up a Glossary Entry

To set up a glossary entry for a ticksheet item, click New in the Glossary Entry area of the dialog box and type the entry name and help text (that displays to end users) in the Glossary Entry dialog box (Figure 31). Click OK.

Click Edit to open a Glossary Entry dialog box for editing.

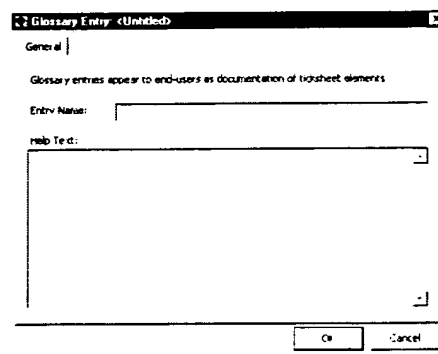


Figure 31: Glossary Entry Dialog Box

Filters Tab

Filters enable ticksheet users to restrict the data accessed for a query to specific attributes (values in dimension columns); for example, the data may be filtered so only the values for direct sales during the years 1990 through 1995 are returned.

You can use the Filters tab of the Ticksheet dialog box (Figure 32) to define the filters for the ticksheet.

After filters have been added, the information is displayed in the ticksheet's Filters window. You can double-click one of these listings to open its dialog box for editing, or select it and click Edit.

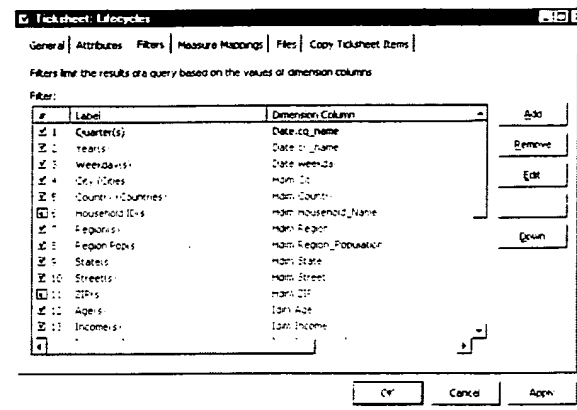


Figure 32: Ticksheet Dialog Box: Filters Tab

Follow these steps to define filters:

Step 1: Click Add to open the Filters dialog box (Figure 33), which has two tabs: General and Filter Elements.

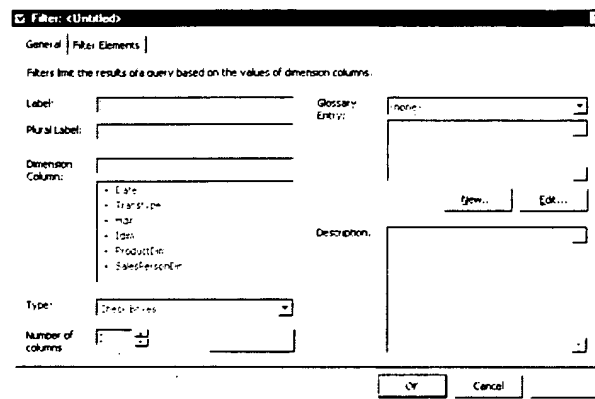


Figure 33: Filter Dialog Box: General Tab

Filters Tab

- Step 2:** In the General tab, enter the name of the filter category as it will appear on the ticksheet, such as Fiscal Year.
- Step 3:** The plural value is the text that follows *All* as in All Fiscal Years on the ticksheet. The system enters this for you. Verify that it is correct.
- Step 4:** Enter any description for your own reference in the Description text box.
- Step 5:** To add a new glossary entry for this filter category, select New and enter a Glossary Entry as described in “Setting Up a Glossary Entry” on page 134, or select an existing entry.
- Step 6:** Select the dimension column for which data will be filtered from the dimension tables tree. Right-click to expand a plus sign. The selected dimension table name and column name are displayed in the Dimension Column textbox.
- Step 7:** Select the type of filter that the user will select in the Filters window of the ticksheet: a check box, a dynamic listbox, list members, or radio buttons, a (static) list box, or a text box. The contents of a dynamic listbox are updated when the Application Server starts. (Dynamic listboxes are refreshed; static listboxes never change.) List members are derived from Momentum ticksheets. (Momentum lists can be used as filter options in Clarity.)
- Step 8:** For the check box filter type, select the number of columns that the attributes will be divided into; for example, Fiscal Quarter has four columns. The filter user interface changes depending on the filter type.
- Step 9:** You can use the Filter Elements tab to create Filter Groups and Filter Elements (see Figure 34).

A *Filter Group* is a logical grouping of filter elements that applies only to check box filter types. For example, a filtering by year may have the Group Q197 (first quarter of 1997), with the months January '97, February '97, and March '97.

A *Filter Element* is a single check box for filtering within a Filter Group, or a single entry within a listbox.

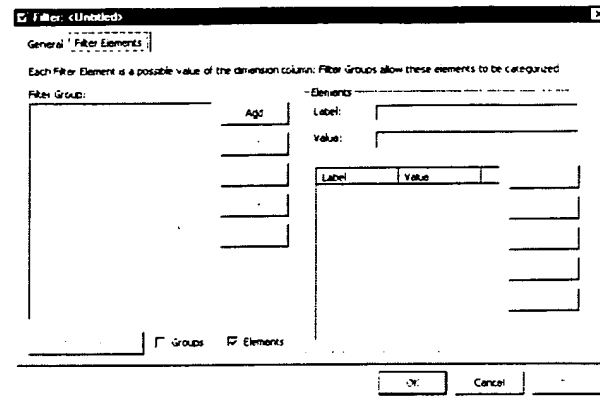


Figure 34: Filter Dialog Box: Filter Elements Tab

Defining Filter Groups

Filter Groups provide a convenience for ticksheet users. Creating a Filter Group does not modify the database. A sample check box Filter Grouping follows:

Group 1 — 1996: Q1 96, Q2 96, Q3 96, Q4 96 (*where Q1 96, Q2 96, and so forth are filter elements*)

Group 2 — 1997: Q1 97, Q2 97, Q3 97, Q4 97

Group 3 — 1998: Q1 98, Q2 98, Q3 98, Q4 98

You can use the Filter Group panel of the Filter Elements tab to define a Filter Group. Follow these steps:

Step 1: Click Add. The Filter Group dialog box is displayed (see Figure 35).

Step 2: Enter the label name (for the group name on the ticksheet) and any description for your reference.

Step 3: Click OK.

Filters Tab

Step 4: To add the filter group as a glossary entry, click New. Instructions are given in “Setting Up a Glossary Entry” on page 134.

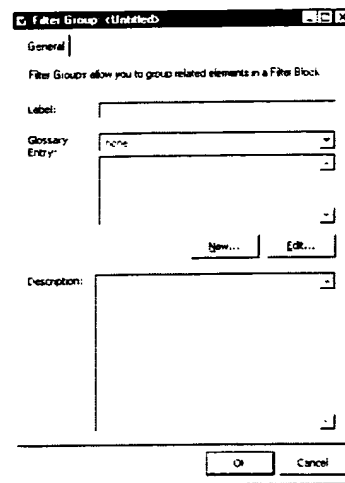


Figure 35: Filter Group Dialog Box

Defining Filter Elements

When entering filter elements, note that there is both a label and a value for each element. The value is the actual database value that will be filtered, whereas the label is what appears to the end user on the ticksheet *and* the report page. If the value *CA* has a label of *California*, then when the user clicks the *California* checkbox, the database query actually filters on *CA*. When the results are displayed, *CA* is replaced with *California*. This feature allows database values to be mapped both on the filter and display side.

To define Filter Elements for the group you created, follow these steps:

Step 1: Open the Filter Elements tab of the Filter dialog box.

Step 2: In the Elements panel, click Add. The Filter Elements dialog box (Figure 36) is displayed.

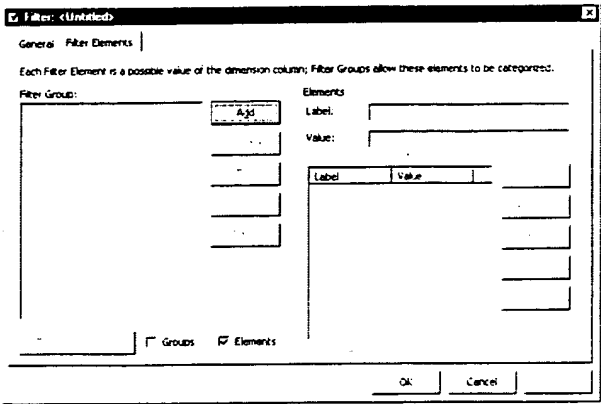


Figure 36: Filter Elements Dialog Box

Step 3: To have the system fill in the list using an SQL SELECT statement (to extract a field from a table), click Fill from Query. The SQL Query dialog box (Figure 37) is displayed.

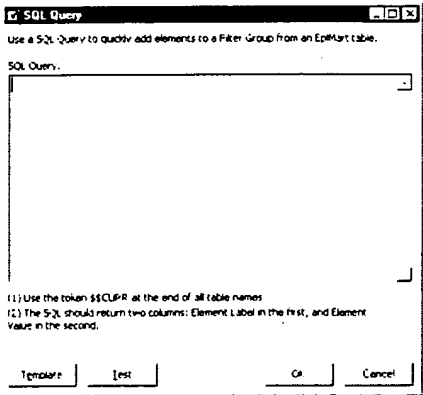


Figure 37: SQL Query Dialog Box

Measure Mappings Tab

- Step 4:** To have EpiCenter Manager create a template SQL to automatically fill your filtering query, click the Template button.
- Step 5:** Replace the column name and the table name for your data.
Append the token `_0$$CURR` at the end of all table names; for example, `Product_0$$CURR`.
- Step 6:** Click Test to see the query that this produces. This query has two columns by default: one for the element's label and another for the element's value.
- Step 7:** Click OK to begin the actual extraction.

Measure Mappings Tab

Note: Measure Mapping does not apply to Relevance Influence ticksheets. Relevance Influence uses measure sets, which are discussed in “Measure Sets” on page 156.

Measure mapping is a two-step process. First, you will add all of the measure column elements that will appear on the final ticksheet and arrange them by column. This sets up the organization of the Web page ticksheet and assigns names to the measure column elements in the ticksheet (the actual value is derived from the Measure to which you map this combination of elements).

Second, you need to map each set of measure column elements that a user might choose from each column in the ticksheet to the measure calculation that this choice represents.

The Measure Mapping tab also allows you to create a new Measure. Click New Measure, which opens the Measure dialog box (see Figure 25). Follow the instructions given in “Measures” on page 118.

Adding Elements to Columns

To add elements in the appropriate columns:

Step 1: Open the Measure Mappings tab (Figure 38) in the Ticksheet dialog box. Note that a single empty column is displayed for a new ticksheet.

Step 2: Click Add to add the measure column elements for this column.

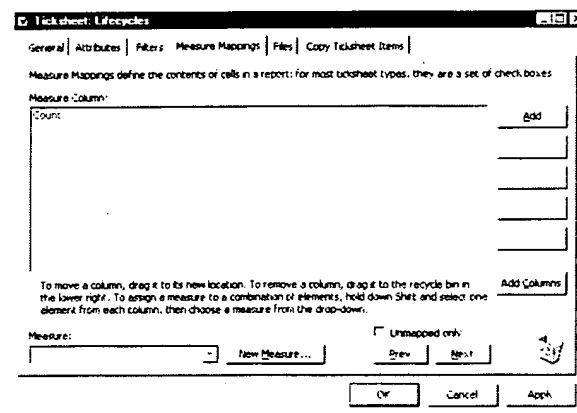


Figure 38: Ticksheet Dialog Box: Measure Mappings Tab

Step 3: You can use the Measure Column Element dialog box (Figure 39) to define the elements. Enter the label (the name as it will appear in the ticksheet column, an abbreviation (if appropriate), and a description for your reference.

Measure Mappings Tab

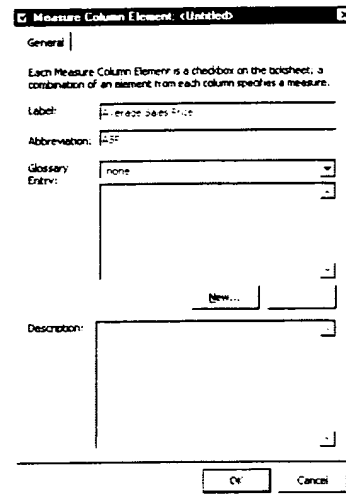


Figure 39: Measure Column Element Dialog Box

Step 4: To set up a glossary entry, click New and type the entry name and help text (that displays to end users) in the Glossary Entry dialog box (see Figure 31).

Measure column elements in ticksheets, such as Units, Gross, and Sell-Through, can be hyperlinked to a glossary page. When a user clicks a link, a glossary page displays which defines it. These glossary entries help users to understand your terminology.

Step 5: Click OK to add the measure column element to this column.

Step 6: Click the Add button again to add another measure column element to the column. Continue to add elements as described above until you have entered all of the elements for the column.

Step 7: Click Add Columns to add a second empty column. Click Add to enter the new column's measure column elements as described above.

Step 8: Click Add Columns to add a third empty column, if applicable. Add the measure column elements to the column. Add additional columns and measure column elements as appropriate.

Step 9: To change the order of one column with another (the order in which they display on the ticksheet), select a column and drag it onto the new column location. The columns' contents are switched.

Step 10: To remove a column, drag it to the recycle bin in the lower right of the dialog box.

Mapping Elements to a Measure

Users can select any combination of measure column elements (one per column) on a ticksheet. Each combination of elements equals a measure (whose calculations determine the contents of the generated report or query). For each ticksheet, you will need to map every measure column element in a given column to every other measure column element in all of the other columns (to cover all the possible combinations that a user may select).

To map elements to a measure:

Step 1: While holding down the Shift key, select an element from each column. Each selection is highlighted.

Step 2: With all of your selections highlighted, choose a measure from the drop-down list.

This measure will be invoked by the system whenever the user selects this combination of elements on the ticksheet.

Verifying that All Elements are Mapped

Click the Previous and Next buttons to cycle through the combinations of measure column elements that comprise a measure. The mapped element in each column is highlighted, and the measure that these elements map to is shown in the Measure text box.

Reports Tab

Select Unmapped Only in the Measure Mappings tab to display only those combination of elements that have *not* been mapped. (Remember that each distinct path through the columns must be mapped.)

Click the Previous and Next buttons to cycle through the possible combinations. Unmapped elements are highlighted, and no measure name is displayed in the Measure text box.

Reports Tab

The Reports tab of the Ticksheet dialog box lists all reports saved for this ticksheet by report name, Report Gallery folder, date last modified, and the user who modified it.

Important: Whenever a new ticksheet is created, the administrator should save one report in the Default subfolder in the Report Gallery's Public folder (see "Report Gallery" on page 209) and make it the default. This allows check boxes to be set to reasonable values for all users when they first use the ticksheet.

You may select a report and click Remove to delete it. (If appropriate, you can delete the entire list.) You may want to delete "orphaned" reports, which are reports that cannot be accessed by anyone via Epiphany's front-end Application Server, and may occur when a user is deleted. Select the Only Show Orphans option in the dialog box to display orphaned queries for deletion.

Select a report in the list and click Edit to display the Report dialog box for this ticksheet for editing. (Figure 15).

Configuring Relevance Ticksheets

Preserving existing items does not work when you copy Measure Mappings. If, however, you are copying attributes, filters, and measure mappings, and *Preserve existing items* is selected, you are warned that the measure mappings will not be preserved. Then all of the other items are copied with preserve and the measure mappings are copied without preserve.

When the system copies measure sets (Influence ticksheets only), it attempts to link them to the same attributes as on the source ticksheet. If there is not one with the same name, it copies the set, but does not link it to any attribute.

To copy items from this ticksheet to another ticksheet:

Step 1: Select from the Copy To: box any ticksheet that you want to inherit items from this ticksheet.

Step 2: Select the items to be copied to these ticksheets.

Step 3: Click Go to begin the copy.

CONFIGURING RELEVANCE TICKSHEETS

The configuration of Relevance ticksheets, other than Influence ticksheets, (see “Relevance Influence Ticksheets” on page 152) is similar to that of a Clarity ticksheet. If you have created Clarity ticksheets that have similar attributes and filters to Relevance ticksheets, you can use the Copy Ticksheet Items tab of the Ticksheet dialog box (Figure 40, on page 145) to copy attributes and filters.

Each Relevance ticksheet has its own attribute roles. For example, the Best & Worst ticksheet has only one kind of role, a BestWorstDimension, and only attributes assigned this role appear in the ticksheet attributes area.

In general, there is no reason to conceal attributes on a Relevance ticksheet from users. The exception is when there are so many that the ticksheet becomes cluttered, or when an attribute has extremely high cardinality (many values). For queries involving such attributes, the Best & Worst or Influence query may be very slow.

COPY TICKSHEET ITEMS

You may use the Copy Ticksheet Items tab (Figure 40) of the Ticksheet dialog box to copy the attributes, filters, measure mappings, and measure sets (when working with an Influence ticksheet) from one ticksheet to another ticksheet within the constellation. *All* members of the selected items, such as all filters, are copied from the current ticksheet to the target ticksheet. You may, however, copy one or more of the ticksheet items; for example, copy only the attributes, or only the attributes and filters to the ticksheet and leave the measure mappings.

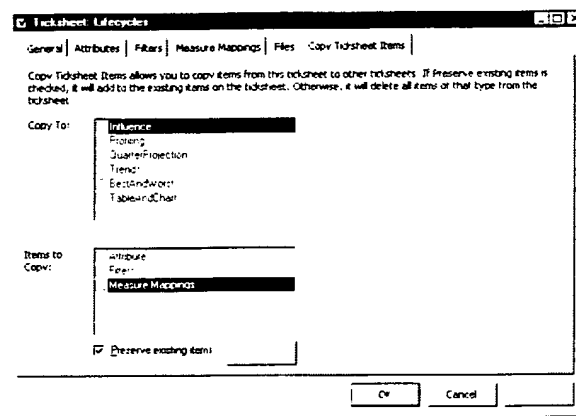


Figure 40: Ticksheet Dialog Box: Copy Ticksheet Items Tab

When *Preserve existing items* is selected in the tab, the copied items are added to the existing ones, creating duplicates on the target ticksheet. Thus if both the source and target ticksheets have a filter named *Channels*, after copying, the target ticksheet has two filters named *Channels*.

Important: If *Preserve existing items* is not selected, the copied items replace (delete) all members of the same item on the target ticksheet. Be sure that you intend to delete these.

Creating Default Relevance Ticksheets

For the Profiles default report, you could select all attributes. If you do not select all attributes, at least attempt to produce a variety of chart types in the output. Note the following:

- Choosing any extremely high-cardinality attribute should generate an 80/20 chart.
- Choosing a high-cardinality attribute should generate a pareto chart.
- Low-cardinality attributes usually result in pie charts.
- A measure for which some attribute values have net negative totals can generate a stacked bar chart.

For the display options, select medium-sized charts.

Choose Auto for the units, and zero digits of precision.

Trends

The *Trends* ticksheet has two roles: TrendsRow and TrendsColumn. The TrendsColumn role should be assigned only to date attributes, such as Fiscal Quarter, Fiscal Year, and so forth. The Trends Row role may be assigned to any non-time attributes.

For a Trends default report, you can use the “straight line fit” with two periods forecasted.

Choose columns and filters such that about four or five time periods of actual data exist. (This way all of the projected columns will fit on users’ monitors.)

For display options, select yes for Include Charts, Auto for the currencies and quantities, and 0 decimal places. Do not select Include Sums. Set Percentages to neither. Select the top 10 rows, all columns, sort rows by amount, and set Filter Current Period to yes.

This last option means that if the last column of real data is an incomplete time period, for example, it is Q3 1998, and the last extract date is September 8, 1998, then Trends realizes that Q3 is incomplete, and will not use it in fitting a trend line to the data.

Creating Default Relevance Ticksheets

When users first use Relevance ticksheets, they may be slightly confused by the many new kinds of analyses they can do. To make their experience with Relevance positive, follow these general rules when creating default Relevance ticksheets:

- The default ticksheet for a Relevance report should convey the value of the Relevance analysis; that is, it should be something that could not be accomplished via a Clarity ticksheet.
- The report should run in 10 seconds or fewer.

Guidelines follow for creating default Relevance ticksheets.

Best & Worst

The *Best & Worst* ticksheet has only the BestWorstDimension role only.

Choose at least three or four attributes. If you choose only two, the user may fail to see the value of Relevance over Clarity for this type of query. Best & Worst excels at analyzing multiple attributes.

Keep in mind when configuring a default Best & Worst ticksheet that end users can readily understand the comparison method “compare to another filter group.” They can look at reports that compare the most recent complete time period (for example, Q3 1998) with the previous complete time period (Q2 1998). In this case you would set the comparison filter to Q2 1998, and set the main filter to Q3 1998. For display options, you can select the top and bottom seven results, which shows enough data without being overwhelming. Choose the “one at a time” option to make the query run faster.

Try various attributes and evaluate the results in terms of an effective report.

Profiling

The *Profiling* ticksheet also has only one role, the ProfilingDimension. Only attributes assigned this role will appear on the ticksheet. Profiling is not affected by high-cardinality attributes as much as Best & Worst, so avoiding clutter is the main reason to not assign attributes to the ProfilingDimension role.

Aggregates for Relevance

Aggregates for Relevance

For the most part, Relevance can use the same aggregates that would normally be built with Clarity. Some parts of Relevance, however, need Relevance-specific aggregates to prevent degraded performance. This section discusses the queries that are constructed when running Relevance, and the aggregates that should be built to make these queries run faster.

Quarter Projections

Typically, creating a Quarter Projection ticksheet involves constructing a new attribute that is not used elsewhere in the system, usually “Weeks til end of Quarter.” No aggregate exists on this attribute, and thus all Quarter Projection queries will access the base dimension table, and take an extremely long time to run. Build an aggregate on the QuarterProjectionColumn and QuarterProjectionRow dimensions, in addition to a few other dimensions that might be commonly used as filters for Quarter Projections reports.

Profiling

Profiling queries involve a single attribute at a time. Ideally, each attribute used in the profiling ticksheet has its own aggregate, or it is included in other small aggregate tables.

Often it can be valuable to include high-cardinality attributes such as Customer or Product on a profiling ticksheet. For Clarity, you probably would not build aggregates involving these attributes. Clarity would never use one of these attributes by itself, but rather in combination with one of many other attributes, and the resulting aggregates might be too large. Because Profiling looks at individual attributes, however, it can make sense to build an aggregate even on a high-cardinality attribute.

Quarter Projections

The *Quarter Projections* ticksheet has two roles: ProjectionRow and ProjectionColumn. The attributes for ProjectionRow should be time attributes only, such as Fiscal Quarter or Month. The attributes for ProjectionColumn should be relative time attributes only, such as Days until End of Fiscal Quarter, or Weeks until End of Fiscal Quarter, or Days until End of Month.

When creating a default report for Quarter Projections, follow the 10-second rule. This usually means building a special aggregate that includes Fiscal Quarter and Weeks til end of Fiscal Quarter, or possibly adding Weeks til end of Fiscal Quarter to an existing aggregate that includes Fiscal Quarter. You can use the same default options as in Trends.

Lifecycles

Lifecycles ticksheets are similar to Trends ticksheets. The LifecyclesColumn role should be assigned only to date attributes. The LifecyclesRow roles may be assigned to any non-time attribute.

Note: The columns in Trends and Lifecycle ticksheets should be absolute date columns, such as the fiscal year or the month of a particular year, such as January 1998. They should not be relative or cyclical columns such as month (that is, January, February, and so forth).

Creating an effective Lifecycles default report may take a little more work than the other Relevance ticksheets. Choose an attribute for the rows, such as Product. You may want to set filters so that there are not too many rows; for example, by filtering on Product Line so that the rows contain mostly different versions of the same product. Set the options similar to the way you would in Trends.

Create the report, and note the beginning dates for the rows. Any row whose beginning date is the same as the first time period included in the filter settings, or whose first time period is the first time period that data exists in the Epiphany system, is most likely already in the middle of its lifecycle, and should be filtered out. (Select all rows and then de-select those rows that were already in the middle of their lifecycles in the time period the report is analyzing.)

RELEVANCE INFLUENCE TICKSHEETS

Creating an Influence ticksheet is similar to creating Clarity and other Relevance ticksheets. All of the ticksheet tabs are set up in the same way. Influence has a specific Measure Sets tab, and no Measure Mapping tab.

How Influence Works

To answer end-user queries, the Influence application builds models that use two types of trees: classification trees and regression trees. A *classification tree* finds rules that can predict the value of a particular discrete-valued *attribute* based on the values of a set of other attributes, and a *regression tree* finds rules that can predict the value of a particular numeric *measure* based on the values of a set of attributes.

Classification Trees

You can use Influence to answer questions such as “Given the attributes of an individual, predict whether that individual will respond to a marketing campaign,” or “Given the attributes of a customer, find the customer segment that customer belongs to.” Influence answers these kinds of questions by building a model—a classification tree—that tries to predict the value of a particular attribute based on the values of a set of other attributes. This is termed *classification* because Influence attempts to predict which class someone or something will belong to; for example, it attempts to classify individuals as responders or non-responders, or as Tier 1, Tier 2, or Tier 3 customers.

The models built by Influence can also provide the answer to related questions about the importance of certain attributes. An Influence classification tree that classifies individuals based on how likely they are to respond to a marketing campaign also provides answers to the question, “Which attributes of an individual have the most influence on whether that individual will respond to a marketing campaign?”

Influence

Since Influence is used to discover the factors that most affect particular aspects of a business, aggregates should be built based on these business entities (that is, the primary dimension, such as customers or products).

First, there needs to be a fact table constructed that contains one row per business entity (denoted by the inclusion of a key to this entity). Each such row should also contain both the attributes that appear on the Influence ticksheet, as well as various measure sets (that is target variables). For example, there may be one row per customer in such an aggregate, and the columns could denote various demographic attributes, as well as various target variables, such as whether or not the customer bought particular products. Note that this may be a large table.

Influence will also make use of aggregates built on the table described above. Such aggregates should include columns such as the primary dimension key, some subset of the ticksheet attributes that seem most likely to affect the business and the measure sets of interest. For more information about Relevance, see “Relevance Influence Ticksheets” on page 152.

Best & Worst

Best & Worst has two methods for analyzing data. The first method looks at attributes individually. In this case, Best & Worst can greatly benefit from having single attribute aggregates (similar to the case of Profiling). The second method looks at combinations of attributes. Here it is useful to have aggregates that contain subsets of related attributes. For example, Education and Income may be highly correlated demographic attributes and should probably be included in the same aggregate.

Trends and Lifecycles

Both of these applications use aggregates that you have probably already built for Clarity, since they essentially run a restricted kind of Clarity report (where the column is always a date dimension and the rows are always non-date dimensions) and then do some post-processing.

Setting Up an Influence Ticksheet

Determining the Primary Dimension

The *primary dimension* is the dimension role that represents the entity about which Influence will make predictions.

The first step in configuring an Influence ticksheet is to determine the *single* primary dimension. (Each Influence ticksheet must have one and only one primary dimension, although it may have multiple targets and multiple source attributes.)

Note: When Momentum is also installed, the primary dimension for Influence is often identical to the Momentum individual or household dimension.

You do not have to explicitly specify the primary dimension in EpiCenter Manager. The choice of the primary dimension, however, restricts the allowable targets and source attributes.

Determining the Target

The target of an Influence query is the *value* that the Influence model attempts to predict. You will define targets in EpiCenter Manager using the Measure Sets tab (see Figure 42) of the Influence ticksheet dialog box.

A *measure set* is a named collection of measures. Measure sets are used to define the target variables you care about predicting when using Influence. A measure set includes the target attribute you would like to predict, as well as a measurement on that attribute. All measure sets include a Count measure, and most measure sets additionally include either a Sum measure or an attribute. See “Measure Sets” on page 156.

Determine the Source Attributes

Source attributes are the attributes whose values Influence uses to predict the value of the target for each member of the primary dimension.

Regression Trees

You can also use Influence to answer questions such as “Given the attributes of a customer, predict the total amount of revenue that customer will generate.” In this case, the prediction is a measurement relating to the customer, not an attribute. The model that Influence builds to answer this type of question is known as a *regression tree* because it performs regression in the statistical sense; that is, attempts to predict the value of one variable (total revenue) given the values of others (the attributes of the customer).

Within Epiphany, the difference between classification and regression is that classification trees use attributes to predict an attribute, and regression trees use attributes to predict a measure. The basic questions answered by the two types of trees are very similar (as are the techniques used by Influence to build them), but they require slightly different configuration.

Setting Up an Influence Ticksheet

Before creating an Influence ticksheet, determine:

- Step 1:** Whose behavior you want to predict; this is the *primary dimension*.
- Step 2:** What you are trying to predict about the primary dimension; this is the *target*.
- Step 3:** The attributes needed to make the prediction; these are the *source attributes*.

For example: “How do the *age*, *income*, and *city of residence* of a customer influence whether that customer will *buy an extended warranty*?” The primary dimension is Customer, the target is the attribute Bought Extended Warranty, and the source attributes are Age, Income, and City of residence.

Setting Up an Influence Ticksheet

Measure Sets

A measure set consists of selected measures that have an assigned name. Each measure within a measure set is also assigned the measure role of Count or Sum. The meaning of a measure role depends on whether the measure set is used for classification or regression.

To define a measure set for classification (a classification target):

Step 1: Open the Measure Sets tab of the Influence ticksheet and click Add.

Step 2: Click the New button in the Choose Measure dialog box. The Measure Sets dialog box is displayed (Figure 41).

Step 3: Enter the name of the measure set.

Data Store	Role

Figure 41: Measure Set Dialog Box

Step 4: When defining a measure set for a list membership target (that is, a measure set with no attribute and only the Count role defined), you can enter a short description in the Description text box that will show up in the Influence ticksheet target selection box. This is useful for cases in which the list membership target is weighted by a measure that is not the count. For example, if the user associates Revenue with the Count role,

For any row in the primary dimension (for example, an individual), there must be a unique value of any source attribute. Obviously, for attributes from the same dimension, there will be only one value. For attributes from another dimension, care must be taken to ensure that this relationship holds. For example, if the primary dimension is individual, one would not choose Product as a source attribute because individuals can buy many products, so it would not be the case that for any individual there would be a single value of the product attribute.

When the primary dimension “rolls up” to another dimension role, source attributes from that dimension role are also acceptable. For example, suppose there is an Individual dimension role and a Household dimension role. If the primary dimension is Individual and no Individual belongs to more than one Household, then Individuals “roll up” to Households and attributes from the Household dimension role can be used as source attributes. In general, source attributes that are not selected from the primary dimension must not have a one-to-one or one-to-many relationship to the primary dimension.

Important: When configuring an Influence ticksheet that has source attributes that are not from the primary dimension, ensure that, because of the properties of your data, the appropriate relationship between source attributes and the primary dimension holds. EpiCenter Manager does not enforce the requirement of a one-to-one or one-to-many relationship, but if this type of relationship does not exist, then some or all Influence queries may fail.

You will use the Attributes tab of the Influence ticksheet dialog box to select the source attributes from a list of the existing attributes. (If appropriate attributes do not exist, you need to create them as you would any other attribute.) Each source attribute must be assigned the TreeDimension attribute role. Attributes assigned to this role are available to the user for inclusion in the set of attributes whose values are used as input data for classification or regressions.

In some cases, you may also want to include attributes on your Influence ticksheets that are not assigned to any attribute roles. For example, to give users the option to build a classification tree to predict the value of an attribute that is not allowed to be used as one of the set of predictor attributes. These attributes need to be on the Influence ticksheet, but should not be assigned to the TreeDimension attribute role.

Setting Up an Influence Ticksheet

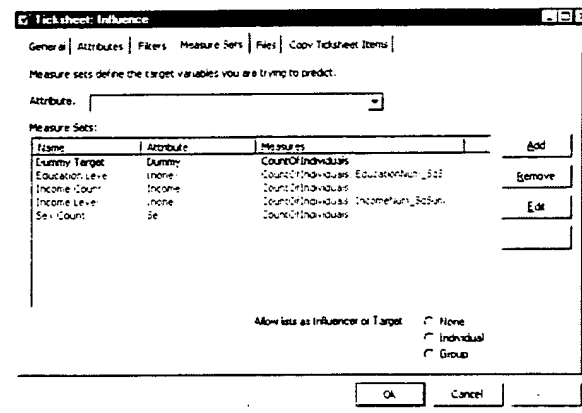


Figure 42: Influence Ticksheet Dialog Box: Measure Sets Tab

Setting Up Targets for the Ticksheet

After defining measure sets, you need to set up classification and regression targets for the ticksheet:

Step 1: In the Measure Sets tab, select a measure set from the list.

Step 2: *For classification targets only:* select an attribute to be associated with this measure set from the Attribute drop-down list.

The attribute whose value is being predicted *must* belong to the primary dimension. The measure you select for the classification target's measure set should give a weighted count of the rows in the fact table.

Note: Regression targets are not associated with an attribute because the value being predicted is a measure value rather than an attribute value.

Step 3: Clicking Update sets this attribute as the target for the selected measure set and makes it available to the ticksheet.

Step 4: Configure the other Influence ticksheet tabs following the instructions given for setting up Clarity and Relevance ticksheets.

then he or she may want to enter “weighted by revenue” in the description box of the measure set. Then, the target will show up in the Influence ticksheet as Member of List, weighted by revenue.

Step 5: Select Count from the Measure Role drop-down list. (The Sum measure role is not used for classification.)

Step 6: Click Add Measure and choose a measure from the list that gives a weighted count of the members of the primary dimension. (This measure must have already been created using the Measure dialog box as described in “Defining a Measure” on page 118.)

For example, choose a measure that computes a simple count of the number of units shipped, or a measure that sums the price of all products shipped, which produces a count that is weighted by price.

Step 7: Click OK.

Step 8: You must associate an attribute with this target (see “Setting Up Targets for the Ticksheet” on page 158).

Defining a measure set for regression (a regression target) is similar to defining a classification target. When you select a measure role for a measure set:

- Select a measure for the Count measure role that counts the number of members of the primary dimension.
- Select a measure for the Sum measure role that is the sum of the fact column that corresponds to the fact value.

The Measure Sets tab (Figure 42) allows you to specify targets for classification or regression by choosing measure sets and, optionally, associating them with attributes. Influence decides whether a measure set is for classification or regression based on whether there is an attribute associated with the measure set or not.

Using Momentum

Influence and Slowly Changing Dimensions

When the primary dimension of an Influence Ticksheet uses the Slowly Changing Dimension semantic type, a single logical member of the dimension (such as a single individual) may appear as multiple rows in the dimension table if the attribute values of that dimension member have changed over time (for example, an individual got married or moved to a different state).

In this case, Influence treats each row in the dimension table as a different member of the dimension when building its model. This may occasionally lead to minor differences in counts between Influence and Momentum, but should not present any problems.

Influence and Aggregates

If you build an aggregate on the primary dimension of an Influence ticksheet that includes all columns of the base dimension (or at least all columns that are used as attributes or filters on the ticksheet), Influence queries almost always utilize it.

Performance Issue

Two kinds of queries cause Influence to adopt a more complicated query plan: queries involving membership in Momentum lists as source attributes or targets, and regression queries that are filtered on dimensions other than the primary dimension. All other things being equal, these kinds of queries tend to take somewhat longer than ordinary Influence queries. If the processing time for Influence queries is a concern to your installation, you may want to avoid these two types of queries.

USING MOMENTUM

Momentum allows end users to generate lists drawn from demographic data for groups and individuals, such as Customers, Resellers, Sites, Households, Individuals in Households, Contacts, or other similar entities. (Note that the terms *groups* and *households* are synonymous in Momentum.)

Tips for Using Influence

Some tips related to using Momentum lists, slowly changing dimension, and aggregates with the Influence application follow. There is also a suggestion for avoiding longer query time.

Using Momentum Lists as Influence Attributes

When the primary dimension of an Influence ticksheet is the same as the Momentum individual or household dimension, Influence can be configured to allow the user to create lists of individuals or households in Momentum and then treat membership or non-membership in a Momentum list as a binary attribute for the purpose of Influence queries.

To specify whether individual lists, household lists, or no lists at all are allowed on a particular Influence ticksheet, use the *Allow lists as Influencer or Target* option on the Measure Sets tab (Figure 42).

For example, you could create a list of all the individuals who bought Product X in the last fiscal year and use that list of individuals as a source attribute in Influence to find out whether buying Product X is a good predictor of some other attribute of the individual (such as buying a related Product Y).

Membership in a Momentum list can be used as either a source attribute or a target attribute for a classification tree. Allowing individual lists to be used as attributes is a legitimate choice *only* when the primary dimension of the Influence ticksheet is individuals, and allowing household lists is a legitimate choice *only* when the primary dimension of the Influence ticksheet is households. You can always disallow lists from being used as attributes.

To add the option of using list membership as a classification target to a ticksheet that has Individual or Group selected for the *Allow lists as Influencer or Target* option in the Measure Sets tab, create a measure set that has only the Count measure role defined (not Sum), as with a regular classification measure set. Do not associate the measure set with an attribute. Measure sets that do not have the Sum measure role defined and are not associated with an attribute are implicitly associated with membership or non-membership in a Momentum list specified by the end user of an Influence ticksheet.

Using Momentum

Step 3: Select the base dimension table that you intend to use for Momentum group demographic data from the list of your base dimension tables. The Momentum constellations folder that is displayed has sub-folders named Facts, Dimension Roles, and Ticksheets.

Step 4: Open the Facts folder, which has one fact called *Ind_Group_Joiner*. This is a special fact table for Momentum that serves to connect the individual and group base dimension tables. It does not function in the same way as other fact tables in the Epiphany system.

Although you cannot modify this dialog box, you do need to define how to extract data into the *Ind_Group_Joiner* table. For more information, see “The Ind_Group_Joiner Fact Table” on page 164.

Step 5: Open the *group* Dimension Role dialog box. Select the Momentum tab and assign a label and a plural name for this demographic dimension. These are the names that appear on the Momentum ticksheet for end users.

Step 6: If you plan to work with individual demographic data and have set up a corresponding base dimension table, right-click the Dimension Role folder and select New from the pop-up menu. A Dimension Role dialog box for the role *indiv* is displayed. Select this base dimension table from the drop-down list.

Step 7: Select the Momentum tab and enter the label and plural names for the individual entity that will appear to end users.

Step 8: The Options Labels tab of the Configuration dialog box has option names and labels that configure special words that display to end users on Momentum ticksheets. See “Option Labels” on page 290 for instructions on customizing these items.

Before you can configure a Momentum ticksheet, you need to set up one base dimension table that will contain the group demographic information for Momentum. Configure this base dimension table in the same way that you would any other base dimension table (see “Base Dimension Tables” on page 98).

As an option, if you want to configure ticksheets that return information related to individuals, in addition to group demographics, define a demographic base dimension table for this individual data. For example, in a fund-raising campaign, you may have data related to households that contribute, in addition to data for the individual contributor. In this case, you would set up two base dimension tables: one for group and one for individuals.

Each EpiCenter may have one Momentum constellation. The Momentum constellation has a built-in dimension role called *group* that points to a single base dimension table that holds group demographic data. The dimension role name for individual demographic data is *indiv*. (The dimension role names cannot be modified.) The *group* and *indiv* dimension roles must point to separate base dimension tables.

After setting up your group base dimension table, and optionally, an individual base dimension table, you are ready to open the Momentum constellation. Mini-dimensions, which are analogous to aggregates, need to be set up in the Column Sets tab of the Base Dimension dialog box. You can set these up later.

Follow these steps the first time you work with the Momentum constellation:

Step 1: Right-click the Constellations folder in the EpiCenter directory tree and select New Constellation from the pop-up menu.

Note: The Momentum Constellation folder will appear in the EpiCenter Manager directory tree only if the Momentum application has been installed, and *Include Momentum* was selected in the Initialize EpiCenter dialog box.

Step 2: In the New Constellations dialog box, select Momentum and click OK.

Step 9: Momentum provides a quick count feature in which you can query a subset of the data, which makes the queries run faster by that factor. You can turn on this feature by assigning a positive value to *min_sample_invlog10*, which is a configuration option in the General tab of the Configuration dialog box (see “*General Settings*” on page 285).

The *min_sample_invlog10* option is the inverse log (base 10) of the sampling probability. For example, a value of 2 would create a 1 percent sample ($1/10^2$).

Notes:

The Ticksheet Types tab of the Configuration dialog box (Figure 82, on page 294) enables you to disable an entire application by choosing the application from the drop-down list and de-selecting Enabled. This is useful if you want to test Clarity and Relevance ticksheets, but have not set up Momentum completely.

As mentioned, the columns you define in metadata are moved into the Momentum tables only if you have selected the option *Can be used in Momentum filter* for dimension columns, and the option *Queryable* for fact columns. Any changes to these options show up after you have run MomentumBuilder, which builds the Momentum tables. (You can run the Scrutiny debugging tool to verify if the Momentum tables are consistent with the metadata; see “Running the Scrutiny Debugging Tool” on page 223).

The Ind_Group_Joiner Fact Table

The Ind_Group_Joiner Fact Table

The *Ind_Group_Joiner* fact table defines the many-to-one relationship between individuals and groups. This table identifies the Momentum demographic dimensions, assigns individuals to groups, and defines the individuals and groups in Momentum. Each of these is discussed below.

- Identifying the Momentum demographic dimensions
MomentumBuilder and the Application Server use the *Ind_Group_Joiner* fact table's special dimension roles *indiv* and *group* to identify the individual and group base dimension tables.
- Assigning of individuals to groups
The data in this fact table determines which individual belongs to which group.
- Defining who exists for Momentum
The table also defines the individuals and groups that are a part of Momentum. Any individual or group that does not have its key in this table does not exist as far as Momentum is concerned. Even if you have individuals that belong to no group or to groups without individuals, you still need to enter their keys in this table for Momentum to recognize them.

How to Populate the Ind_Group_Joiner Table

When populating this table, consider these three cases:

1. Individuals who belong to a group and groups that have individuals. For each individual, assign an entry for the individual and the group that it belongs to. Assign an individual to one group only. This is the most general case and almost all of your individuals and groups will fall into this category.
2. Individuals who do not belong to a group according to the available data. Assign all individuals to the UNKNOWN group.
3. Groups with no individuals. Place an entry for each of the group and for the individual key assign it as UNKNOWN. Now the individual UNKNOWN is a part of multiple groups, but MomentumBuilder is aware of this situation, and it will handle these rows correctly.

Note: In the case of a slowly changing dimension, only the latest attribute is used, unless the attribute comes from a leaf dimension, in which case the attribute at the time of the transaction is used. *Leaf dimensions* are part of adjacent constellations (see “Why Use Adjacent Constellations?” on page 166).

The Momentum Constellation

The constellation you just set up is similar to the one shown in the diagram in Figure 43.

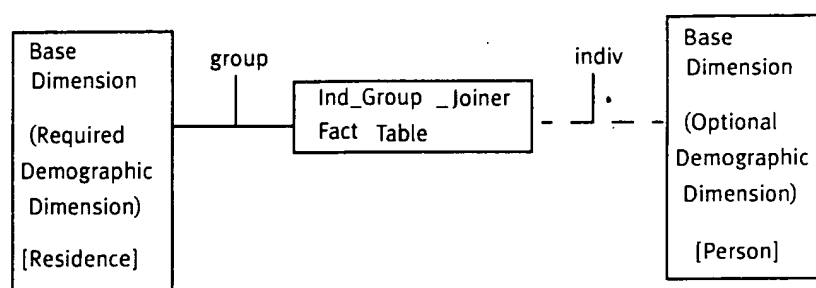


Figure 43: The Momentum Constellation

A user of a Momentum ticksheet applies filters to restrict demographic data; for example, to restrict the list further, or to include others in the list. After the user selects a filter in Momentum, he or she can find out how many matches it corresponds to in the database, either exactly or approximately.

The Momentum constellation shown above allows you to configure a ticksheet that can filter group and individual attribute data. The filtering occurs only on the demographic base dimension tables. Optionally, a Momentum ticksheet can be configured to allow users to filter behaviors, or transactions related to demographic attributes upon which the user has filtered. A *transaction* is usually an action such as bought, returned, called call center, received promotion or responded to campaign.

The Momentum Constellation

To configure a ticksheet that allows users to filter on transactions (that is non-Momentum fact table, such as Order), in addition to attribute data, requires that you set up additional constellations, called Momentum Adjacent constellations. These are optional, but their presence gives you access to Momentum's most powerful list making capabilities.

Why Use Adjacent Constellations?

The Momentum constellation for an EpiCenter is usually associated with other constellations in the EpiCenter, which are called *Momentum Adjacent constellations* (see Figure 44). While Momentum constellations are used for demographic queries, adjacent constellations may be used for behavioral and transactional queries. The information that can be queried is more complex because filtering can occur using Fact tables and non-demographic base dimension tables.

You may set up any number of adjacent constellations in your EpiCenter. Momentum Adjacent constellations can be queried by Clarity similar to any other constellation.

For example, a Momentum query based on a Momentum constellation is capable of returning a list of household members who satisfy a demographic criterion, such as sharing the same Zip code. When an adjacent constellation is also queried, the members of these household who responded to a request for a donation in a certain year could be determined.

Setting Up Momentum Adjacent Constellations

A Momentum Adjacent constellation must have at least one dimension role that points to one of the demographic dimensions (in Figure 44, the *Person role* dimension points to the Person base dimension table). Non-adjacent constellations are constellations that do not have an associated dimension role, and thus are not used by Momentum.

Additional Momentum Configuration

The dimension tables in a Momentum Adjacent constellation that point to non-demographic base dimensions are called *leaf dimensions*. These can be queried by Momentum and filtered via their presence in transactions using standard Clarity-style filtering. In Figure 44, the leaf dimensions are Sales Regions, Sales People, and Products.

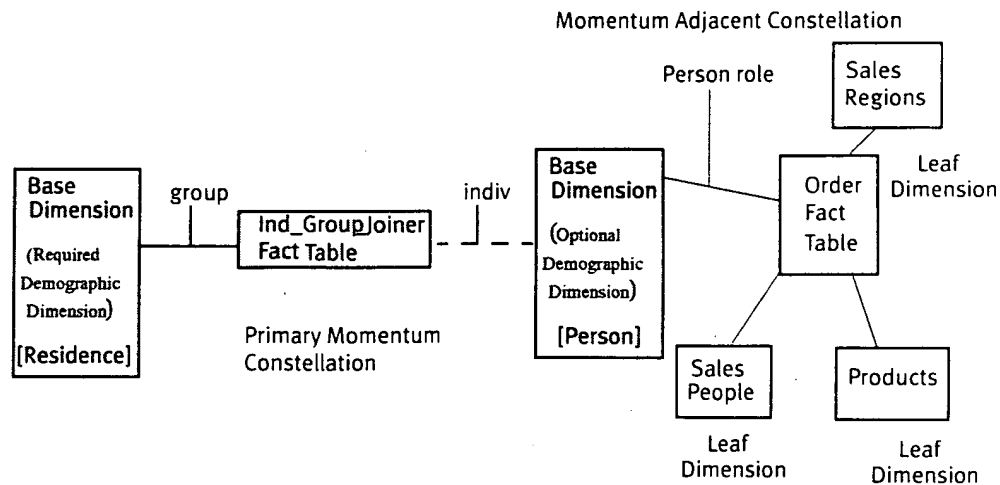


Figure 44: Adjacent Constellation Associated with a Primary Constellation

To set up an adjacent constellation, first set up the Momentum constellation. Then set up a constellation that has the fact table and dimensional attributes that you want to query for demographic data. At least one dimension role must connect the adjacent constellation's fact table to one of the Momentum demographic base dimension tables. In the Figure 43, this dimension role is the *Person role*.

Additional Momentum Configuration

Setting up a Momentum constellation also requires you to—

- define mini-dimensions
- build clusters and counts
- define transaction filters

Mini-Dimensions

Attributes of a dimension can be divided into disjoint sets such that attributes within a set are highly correlated, and there is very little correlation between an attribute from Set A and an attribute from Set B. These different sets of attributes are placed into different tables called mini-dimensions. The mini-dimensions are very small because the attributes are highly correlated, and since the correlation between different mini-dimensions is very low, their join gives a large dimension. Mini-dimensions compress the storage for a table and consequently reduce the time required to bring it up in memory when scanning the table for querying. The storage compression is significant for large demographic dimensions.

Clarity and Relevance have aggregate tables that speed up queries. You define these on the Column Sets tab of the Base Dimension dialog box where you also set up mini-dimensions. The fact that these are discrete allows you to logically separate sets used for Clarity and Relevance from those used by Momentum. There are also other differences between aggregate column sets and mini-dimensions:

- The mini-dimensions for a single base dimension table must not overlap. They are disjoint, whereas column sets may overlap.
- The resulting mini-dimension tables are mapped to integers, which results in mini-dimensions that are thinner than aggregates.

But, most importantly, you design aggregate column sets not only because attributes are correlated in the database, but also because end-user queries need all of these columns. Any single Clarity query can use only one column set. For mini-dimensions you still want data-correlated attributes in the same mini-dimensions, but because a Momentum query can use more than one set, the end-user correlation is not as important.

For example, assume that a customer base dimension table has columns for City, State, Zip, Occupation, Income, and Age. Data correlated attributes are—

Step 1: City, State, and Zip

Step 2: Occupation and Income

Step 3: Age

You can set up mini-dimensions on each of these three divisions. Since users tend to query number 1 and 2 at the same time, you could set up an aggregate on these.

You need to define mini-dimensions both on your demographic base dimension tables, and on the leaf base dimension tables. Mini-dimensions need to be defined in order to set up fact clusters and counts.

Guidelines for setting up mini-dimensions follow:

- Place only queryable attributes in mini-dimensions.
- Group attributes that are highly correlated in real life, such as zip code and area code.
- Place most queryable attributes in a mini-dimension; however, do not place an attribute in a mini-dimension that has high cardinality (many values), such as a name.
- An attribute can appear in one mini-dimension only.
- Because a mini-dimension can hold a maximum of 32,767 rows, create mini-dimensions that are smaller than this size.

Transaction Filters

Assume that a demographic entity, such as People, exhibits a behavior or transaction, such as Bought Product that you want to find out more about. For example, What product did they buy? Whom did they buy it from? When did they buy it? All of these questions can be answered by filtering on the leaf dimensions associated with the Order fact table using regular Clarity-type filters, which you define in the same way as a Clarity filter. The *what*, *who*, *when* types of queries that end-users might have are derived from dimension attributes.

You might also want to know *how much* these people bought, which is a numeric value, or a measure. To enable end users to apply a filter on measure data in this manner, you need to define transaction filters in addition to regular filters. Transaction filters are always associated with a set of regular filters and, optionally, can be configured to apply a measure filter to the result of that filter set.

Clusters and Counts

Defining a transaction filter involves selecting—

- a fact table.
- the appropriate dimension role (the system is flexible and there may be more than one demographic dimension role pointer).
- the transaction type (fact tables can have more than one).

Clusters and Counts

Clusters and counts make transaction filtering faster. A *cluster* is a copy of the fact table sorted on an attribute. A large table should be sorted on disk with its leading term being the most selective filter.

Counts keep statistics about how often the mini-dimension row appears in the fact table. Counts are used by the Momentum query engine to select the best clustered copy of the fact table. (A cluster is useless without a count.)

When you associate a fact table in a Momentum Adjacent Constellation with a mini-dimension, the system creates copies of the fact table clustered on a certain leaf mini-dimension key. Clustering around a mini-dimension speeds up query performance because the fact table rows that need to be accessed are physically contiguous, thereby limiting the number of disk blocks that need to be scanned.

Although clustered copies are very useful, keep in mind that these are copies of the fact tables, and too many of these copies would consume disk space and increase the time required by MomentumBuilder to build the Momentum tables. In contrast, counts are cheap.

For example, users typically ask about buying patterns by customers (demographic dimension) and product lines (attribute leaf dimensions). Define a mini-dimension on Product Line and related attributes, and cluster and count the Order fact table by that mini-dimension.

Guidelines for clusters:

- To be able to cluster a fact on a dimension, you need to specify at least one mini-dimension. You can assign all the queryable attributes for that dimension to a single mini-dimension.

- You need to specify at least one cluster on a fact table for it to be included in Momentum.

CONFIGURING A MOMENTUM TICKSHEET

Similar to Clarity and Relevance ticksheets, you will use the General tab of a Momentum Ticksheet dialog box to name the ticksheet, select the ticksheet type, and assign the ticksheet to a dataset. The ticksheet type is either *Filter Households* or *Filter individual*, and the dataset is always called Momentum (this is built-in).

Setting Up Momentum Attributes

The Momentum attributes are derived from the group or individual demographic base dimension tables.

You will set up attributes for Momentum ticksheets using the Attributes tab of the Momentum Ticksheet dialog box. The only difference between a Momentum attribute and Clarity or Relevance attributes is the kind of attributes allowed. If the ticksheet type is *Filter Household*, then only group attributes are allowed. If the ticksheet type is *Filter Individual*, the attributes may relate to both the group and individuals in that group.

The Role types are *FilterIndividualCols* for *Filter Individual* types and *FilterGroupCols* for *Filter Household* types. These are the columns available for inclusion in the actual lists.

Configuring a Momentum Ticksheet

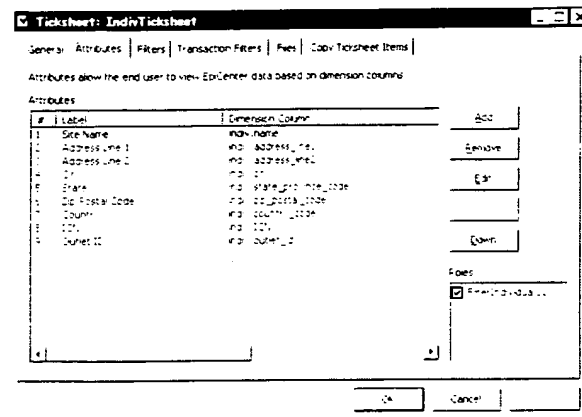


Figure 45: Momentum Ticksheet Dialog Box: Attributes Tab

Filters

All filters for the Momentum ticksheet—demographic filters and those used in conjunction with transaction filters—must be defined in the Filters tab. After filters are defined, they are “stored” here. (Think of this tab as a kind of warehouse of filters.)

Important: In order for a dimension column to be used as a Momentum filter, the *Can be used in a Momentum filter* option must be selected in the Dimension Column dialog box (Figure 17, on page 101).

For demographic filters, if the ticksheet type is *Filter Household*, then only group attributes are allowed; that is dimension columns from the *group* base dimension table. If the ticksheet type is *Filter individual*, then attributes may relate to both the group and individuals.

Transaction filters, which apply only to Momentum Adjacent constellations, are discussed below.

Setting Up Clusters and Counts on Fact Tables

Clusters and counts apply to Momentum Adjacent constellations only. You can use the Momentum tab in the Fact Table dialog box (Figure 46) to set these up. (Mini-dimensions need to have already been created for the leaf dimension tables.) Each selected mini-dimension is a cluster.

To set up a cluster:

Step 1: In the Clusters panel, click Add Dim.

Step 2: Select a mini-dimension from the Choose dialog box, and click OK.

This mini-dimension is added to the cluster list. (The numbering is for internal use only.)

Step 3: Continue to add additional mini-dimensions.

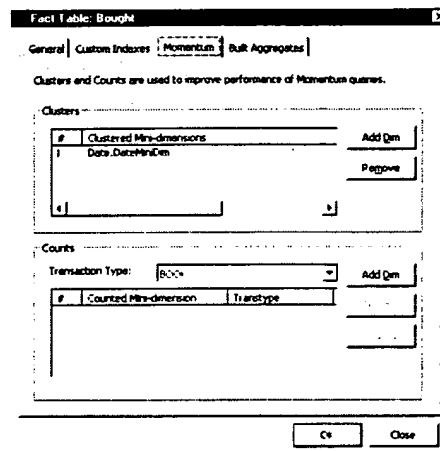


Figure 46: Fact Table Dialog Box: Momentum Tab

Configuring a Momentum Ticksheet

As a rule, if you are building a cluster, also build a count for that mini-dimension. Build Counts on each transaction type that will be queried in the fact table.

To build a count for this cluster:

- Step 1:** Select a transaction type from the pull-down menu to associate with the mini-dimension.
- Step 2:** Click Add Dim.
- Step 3:** Select the mini-dimension to be counted from the Choose dialog box, and click OK.

Defining Transaction Filters

You can use the Transaction Filters tab (Figure 48) to define transaction filters.

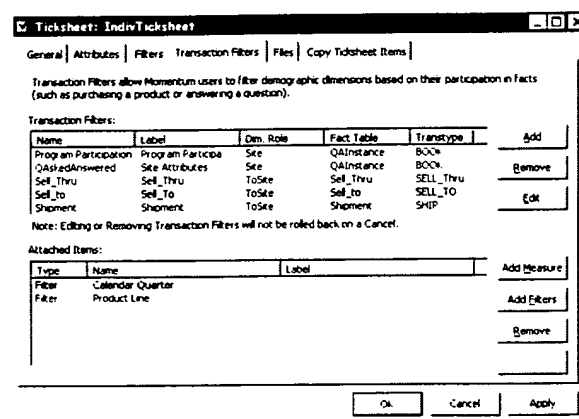


Figure 47: Momentum Dialog Box: Transaction Filters

To define a transaction filter:

- Step 1:** In the Transaction Filters panel, click Add, and click New (or select an existing filter).
- Step 2:** For a new transaction filter, using the Transaction Filter dialog box, enter the name and the label as it should appear on the ticksheet. You may also enter a description for your reference.

- Step 3:** Select the name of the adjacent constellation.
- Step 4:** Select a dimension role that connects the adjacent constellation to the demographic base dimension table.
- Step 5:** If there are multiple fact tables in the adjacent constellation, select the fact table associated with the dimension role.
- Step 6:** Select the transaction type. A fact table may contain multiple transaction types, but a transaction filter applies to only one.

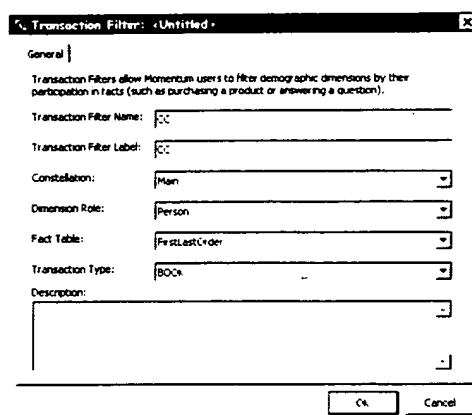


Figure 48: Transaction Filter Dialog Box

To associate a measure in the constellation with a transaction filter:

- Step 1:** Select the filter in the upper panel of the Transaction Filters tab.
- Step 2:** In the lower panel of the Transaction Filters tab, click Add Measure.
- Step 3:** Either select the measure from the drop-down list in the Measure for Transaction Filter dialog box, or click New Measure to display the Measure dialog box, which enables you to create a new measure.
- Step 4:** In the Label textbox, enter the name of the measure as it should appear to the end user on the Momentum ticksheet.

Default Settings on Momentum Ticksheets

The transaction filter is always applied to a set of regular attribute filters. Click Add Filters in the lower panel of the tab and select those filters that comprise the set from the Filter dialog box. Only filters from the same constellation as the transaction filter are allowed.

Default Settings on Momentum Ticksheets

You can specify any configuration of Momentum filters and save this to a default ticksheet in the same way as you would in Clarity or Relevance.

Momentum has two sets of defaults that must be set for a ticksheet: the default settings for the ticksheet itself, which are set the same way as for Clarity or Relevance, and the default settings on the filter pop-up window that displays whenever you create a new filter.

To set the default settings for filters, follow these steps:

- Step 1:** Add all of the default filters that you want to use on the default ticksheet.
- Step 2:** Click the *Restrict list further* link.
- Step 3:** In the filter pop-up window, set all of the default settings.
- Step 4:** Close the window using the Cancel button. (Do not use the Close button.)
- Step 5:** Save the ticksheet as the default as you would a Clarity or Relevance ticksheet. All of the defaults, including the ones in the filter pop-up will be saved for the ticksheet.

EDITING, DELETING, AND DUPLICATING TICKSHEETS

After you have created a ticksheet, you can right-click it and select Edit from the pop-up menu to open its ticksheet dialog box for editing.

You can also use this pop-up menu to delete, duplicate, or export the ticksheet metadata file.

This completes the section on setting up a constellation. To set up another constellation, return to “Setting Up a Constellation” on page 106.

EXTRACTION

After you have configured your EpiCenter, you may use the Extraction folder to set up the extraction process described in Chapter 2. The Extraction folder consists of the sub-folders Data Stores, Extractors, Jobs, and External Tables, each of which is discussed below.

Data Stores

You can use the EpiCenter Manager's Data Store dialog box to create as many data stores as are appropriate for your site. Each extractor has a single input and output data store. Because jobs consist of multiple extractors, a job has multiple input and output data stores. For example, one job might extract data from two source systems into one EpiMart, and another job might extract data from a source system into external tables and from external tables into staging tables. In the second job, the external tables serve as both input and output data stores to different extractors.

The Data Store Dialog Box

Each data store has a Data Store dialog box (see Figure 49), and this dialog box has two tabs: General and Properties.

The General tab has two panels: Data Store Type and Data Flow. The Data Store Type panel of the General tab is where you enter the name and description for the data store and select its type: Microsoft SQL Server, Oracle, Generic ODBC Data Source, or File.

Note: The Logging and Datamart options in the top-right corner are read-only and simply indicate when the opened data store is one of the two special built-in data stores.

The Data Flow panel of the General tab has these options:

- *Allow Use as Input Data Store* allows the data store selected in the Extractor dialog box (Figure 50, on page 181) to be used as the input of an extractor.

The Data Store Dialog Box

The source systems listed in the Source System drop-down list serve to distinguish source system identifier keys that may clash between different database systems. For example, if an account has an ERP and an SFA system, each of which has a Customer table, both may contain a customer number 100. To distinguish between these two records, two different source system identifiers must be used.

Important: Fact rows will join only with dimension rows that have the same source system identifier. For this reason, unless the site actually uses two or more logical source systems, select Datamart Source from the Source System drop-down list (the default).

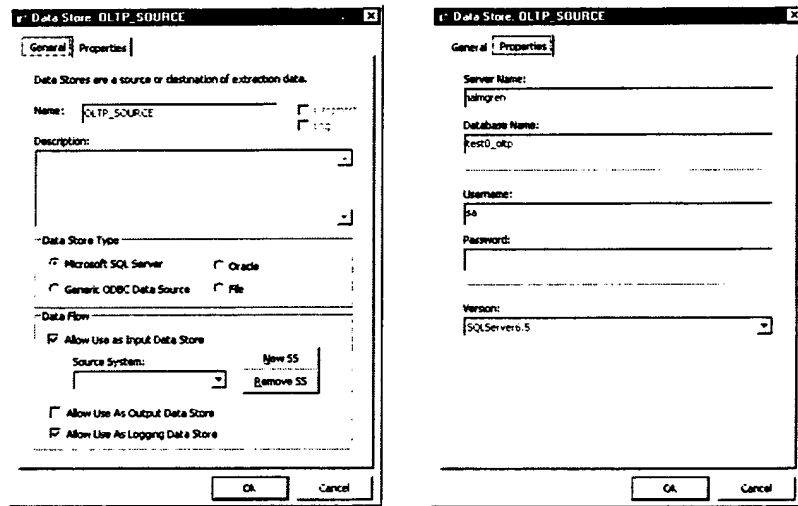


Figure 49: Data Store Dialog Box: General and Properties Tabs

- *Allow Use as Output Data Store* allows the data store selected in the Extractor dialog box (Figure 50, on page 181) to be used as the output of an extractor. De-selecting this option ensures that one does not accidentally write data to a database that is read-only; the data store name does not appear in the list of available output stores for extractors.

- *Allow Use as Logging Data Store*

Epiphany has special metadata tables for the purpose of logging its activity. Normally, this logging is written to the EpiMeta database.

Logging to the EpiMeta, however, may increase its size significantly, so you may prefer to log to another data store. You can use the *logging_mssql.sql* script (included with the Epiphany software) to create a new logging database. If you run this script on another database, then a new data store can be used to log extractions.

Note: The *Allow Use as Logging Data Store* option must be selected in the Data Store dialog box. The purpose of this option is to ensure that logging is directed to the proper logging data store.

The Properties tab has different fields, depending on the data store type. You can use the Properties tab to enter these fields:

- For SQL Server, enter the server's name, database name, username, and server's password and version.
- For Oracle, enter the SQL Net name, username, the server's password, and the version number (Oracle8).
- For a generic ODBC data store, enter the DSN name and ODBC driver name, and server administrator's username and password. (You must also set up a DSN for the data store system using the Data Source Administrator in the Windows NT Control Panel.)
- For a data store of type file (*JobLogFile*), enter its file directory path.

Extractors

Modifying the Default Data Stores

You will need to modify the default data stores, which are *EpiMart*, *JobFileLog*, and *LoggingDB*. Double-click their folders to open them and fill out the dialog boxes as follows:

- *EpiMart* specifies your EpiMart as a data store.
The only value that you can configure is the name of your EpiMart. You can use the Properties tab of the EpiMart Data Store dialog box to enter this database name, for example *Corp_EpiMart*.
- *JobFileLog* specifies the data store for EpiChannel job logs.
In the General tab, the settings are correct for the default usage. The Data Source Type is a file, and the Data Flow is set to *Allow Use as Logging Data Store*.
Click the Properties tab, and change the directory for the *JobFileLog* from CHANGE ON INSTALL to the correct directory name. A valid directory name is required for a successful extraction. Leave the file name blank.
- *LoggingDB* specifies the data store for the EpiChannel logs.
In the General tab, the Data Source Type is your server, and Data Flow specifies: *Allow Use as Input Data Store* and *Allow Use as Logging Data Store*.
You can accept the default values for the log database's file name and directory path, unless you have located the log in a different database.

Note: \$\$DEFAULT refers to the current EpiMeta.

Extractors

In general, SQL statements extract data and semantic instances merge data. The extractor SQL statements may be either stand-alone or extraction (pull/push) SQL statements. Stand-alone statements are typically used to produce a side-effect, such as creating or dropping indexes.

Extractor filters control incremental extraction through the use of special SQL macros.

To define an extractor and its filters:

- Step 1:** Choose New Extractor from the Extraction menu. The Extractor dialog box that displays has two tabs: General and Filter (see Figure 50).
- Step 2:** In the General tab, enter the extractor's name and description.
- Step 3:** Select the appropriate input and output data stores from the drop-down lists. *EpiMart* and all data stores that are enabled for input/output appear in these lists.
- Step 4:** To create a new input or output data store from this dialog box, click the New Input or New Output button, which displays a blank Data Store dialog box for editing.
- Step 5:** Click the Edit Steps button to open the Extractor Steps dialog box (see Figure 51). The buttons on the right side of this dialog box are the means by which you define a new group, new SQL, or a new semantic instance.

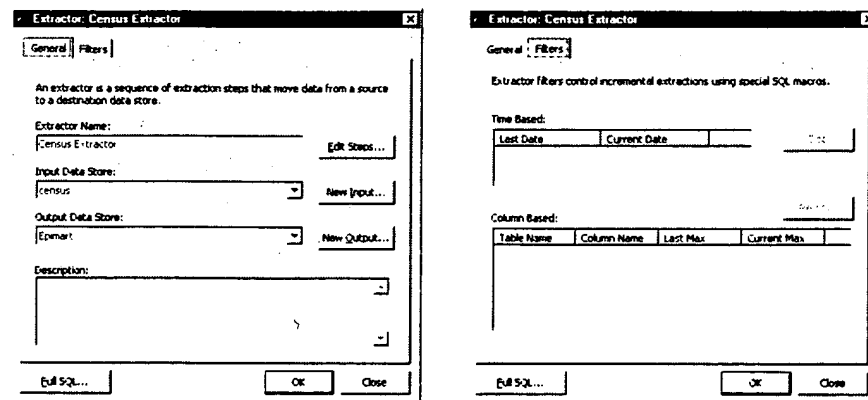


Figure 50: Extractor Dialog Box

Extractors

- Step 6:** In the Filters tab, click the Clear Filters button to clear previous date or timestamp filter memory from the system. For instance, using the `$$TIMESTAMP_RANGE` SQL macro causes the last date of extraction to be “remembered” for this extractor. (See “Extracting New Rows Only” on page 70 for more information.)
- Step 7:** The extraction filter macros are built-in. To modify a macro, click Full SQL and choose its SQL file from the dialog box. Modify the macro and save it with the same file name. This creates a text file with the post translation version of all SQL statements in the extractor (excluding semantics).

The Extractor Steps Dialog Box

Figure 51, on page 183 shows an Extractor Steps dialog box in which extractor steps have already been defined for the extractor. Select the extractor from the drop-down list, or click New to display the Extractor dialog box (Figure 50) in which you can create a new Extractor.

All of the EpiCenter’s extraction groups, which consist of extraction steps, appear in the right side of the dialog box.

When working with the Extractor Steps dialog box, keep these definitions in mind:

- A *job* consists of extractors, which can be shared between jobs. (A job also consists of system calls.)
- An *extractor* consists of extraction groups, which can be shared between extractors.
- An *extraction group* consists of extraction steps that are either SQL statements or semantic instances.

When extraction groups have been defined, you may select the ones for this extractor and move them to the Extractor Steps listbox on the left. Use the arrows to move them over, and the Up and Down buttons to order them in the list.

Because an extractor step must be enabled (checked) to be used in a job, de-select any extractor step that you do not want to be run. You may de-select an extractor step to restart a job that failed, or to debug a subset of the steps. In most cases, if a job fails, you should re-run the entire job.

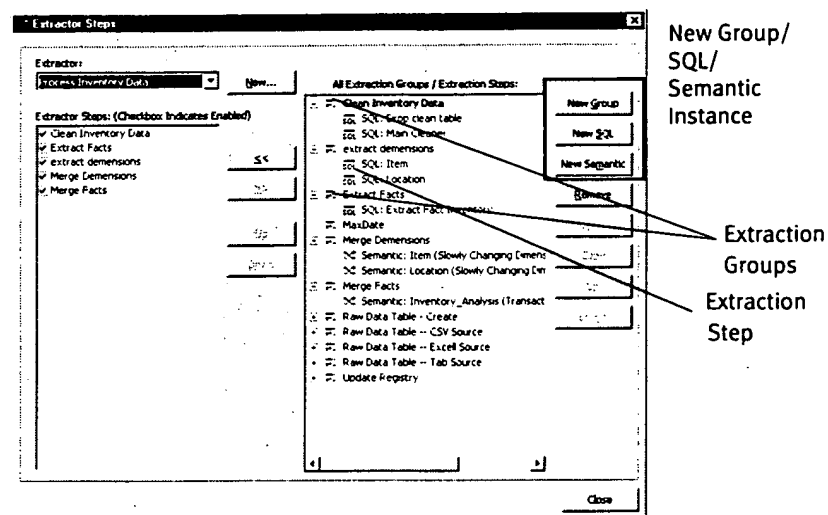


Figure 51: Extractor Steps Dialog Box

Notes:

The *End of Extraction* extractor is provided by default. See “External Tables” on page 50 for a description.

You may right-click a group or an extraction step to display a contextual pop-up menu (for adding a New Group, New SQL, and so forth).

Extractors

Extraction Group

To define an extraction group:

Step 1: Using the Extractor Steps dialog box, click New Group.

Step 2: Enter the group name in the dialog box; for example, *Order Raw* to extract raw data from the Order master table on the source system.

This name appears in the All Extraction Groups/Extraction Steps list in alphabetical order.

Extraction Steps for the Group

Next, you need to define the extraction steps for the group. This group can contain any number or combination of SQL statements or semantic instances. If the step requires SQL, EpiCenter Manager provides a template for sample SQL.

To define SQL for one of the steps in this group:

Step 1: Select its group name in the All Extraction Groups/Extraction Steps listbox, and click New SQL.

The SQL Statement dialog box is displayed (see Figure 52). In the upper panel are the General and Store Types tabs; the SQL and Description tabs are in the lower panel. (You can use the description tab to add a description for your reference.) You can resize this window.

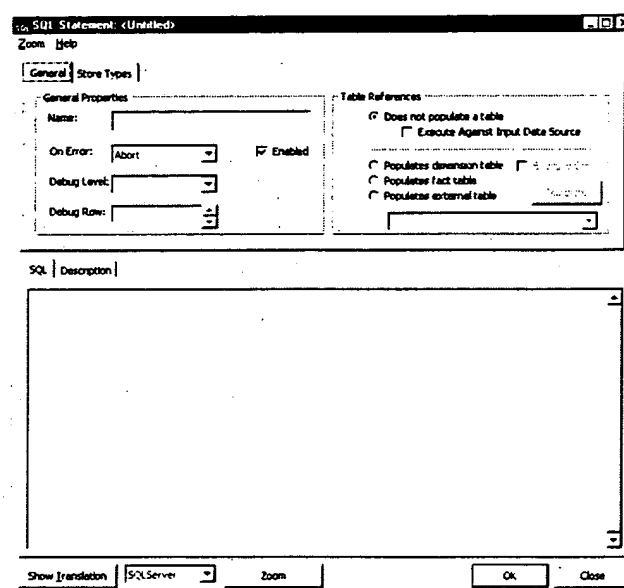


Figure 52: SQL Statement Dialog Box

- Step 2:** In the General Properties panel of the dialog box, enter the step's name (as it will appear in the All Extraction Groups/Extraction Steps listbox). For example, a step named *Customer Raw* extracts raw customer data.
- Step 3:** You can define a step but not have it execute by de-selecting Enabled.
- Step 4:** Select the action to be taken if there is an error with the step. The default is to abort the step. (You can also ignore the error, or have it print to a log and continue.)

Extractors

Step 5: Select the debug level. These levels correspond to the verbosity levels on the **extract** command line. See “EpiChannel Debugging Levels” on page 78.

Step 6: Select the row number at which the debug level you selected above should go into effect, if applicable. If you leave this field blank, the debug level will take effect immediately.

Step 7: In the Table References panel of the SQL Statement dialog box, select whether the step:

- does *not* populate a table (and if this is true, select whether it executes against input data store, or retains its default behavior of executing against the destination data store);

—or—

- populates one of the following types of tables: a dimension table, a fact table, or an external table.

Most SQL statements are used to populate staging tables (or sometimes, external tables). A SQL statement may, however, be used to achieve a side-effect, in which case you will set it to “not populate a table.” In this case, the statement is executed, and any returned results discarded.

Step 8: If the step references a table, select the table from the drop-down list.

Step 9: Execute Only. (*This option is not supported for this release.*)

Select this when SQL is to be expanded to reflect the structure of one of these tables, but the returned rows from the SQL statement are not meaningful.

EpiChannel will not insert rows in the statement.

Extraction SQL is executed against the input data store and must be in the dialect of that database engine. The results are stored in the destination data store.

EpiChannel automatically resolves SQL dialect problems if you use the Epiphany database-vendor independent macros consistently in your SQL. (See Appendix A, “Epiphany Macros” for more information.)

Display Sample SQL

If the step populates a table, clicking the Template button displays sample SQL for the step in the lower panel of the SQL Statement dialog box. The SQL needs to be modified to contain the FROM and WHERE clauses appropriate for the tables in the source system, but the template lists the columns that must be returned. It does not matter in what order the columns are returned as long as they have the proper column names (and “extra” unused columns may be returned).

You can have the system show the how the macros will be translated for your server. Click the Show Translation button and select your server type from the drop-down list.

The SQL Results window displays the SQL in color-coded text. The main menu commands enable you to use standard edit commands to search and replace lines, and you can set bookmarks on lines. The Tools menu has Undo and Print commands, and commands for clearing text with and without clearing the buffer.

The SQL Statement dialog box is re-sizable. You can also click the Zoom button to have the text area fill up the entire SQL or Description window.

Restrict Data Stores for the Extraction

To restrict the input/output data stores for this extraction:

Step 1: Specify them in the Store Types tab of the SQL Statement dialog box.

Step 2: Click Add Type to add a type.

If a type is listed, the statement is disabled for any other type except those listed. For example, you could use the same SQL twice, once with Microsoft SQL Server syntax and once with Oracle syntax, and have one or the other statements automatically disable itself when the source database is of the wrong type. This is of value when the group that contains the SQL statement is shared by multiple extractors. Another approach to handling SQL dialect problems is to use Epiphany SQL replacement macros instead of database vendor-specific constructs. See Appendix A, “Epiphany Macros” for more information.

Extractors

Additional Steps

To create additional steps within the same group, click OK to return to the Extractor Steps dialog box, and repeat the appropriate steps as described above.

Semantic Instance

To set up a semantic instance:

- Step 1:** Either create a new group or select an existing group in the Extractor Steps dialog box.
- Step 2:** Click the New Semantic button. The Semantic Instance dialog box is displayed.
- Step 3:** Select the fact or dimension table that the semantic instance references.
- Step 4:** Select the Object from the drop-down list. An object is either a fact table or a base dimension table (the one to be operated on by the semantic type).

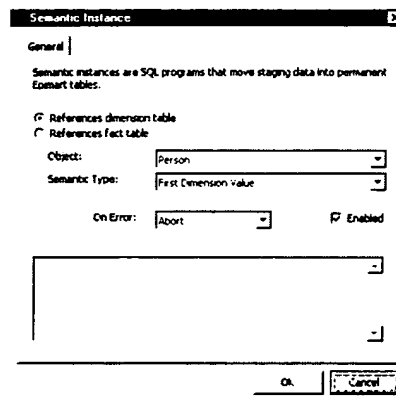


Figure 53: Semantic Instance Dialog Box

- Step 5:** Select the semantic type from the drop-down list. See Appendix F, “Semantic Types” for descriptions of semantic types.

Step 6: Select the action to be taken upon error: abort, ignore, or print. Enabled must be selected for this action to take place.

Step 7: To create additional semantic instances for this group, repeat Steps 2 through 4.

Jobs

The Job dialog box enables you to order the job steps (extractors and system calls) that comprise a job. The Add Job Steps dialog box (available through the Job dialog box) enables you to create system calls.

An EpiCenter has a default job consisting of these default job steps: the Aggbuilder system call for aggregation, the End of Extraction extractor, and the Refresh system call for refreshing the Application Server. (For the Refresh system call to work, the Application Server must be configured as described in the *Epiphany Installation Guide*.) If Momentum is installed, the MomentumBuilder system call is also included as a default job step.

You may customize the default job and create as many other jobs as necessary.

To open the default Job dialog box, double-click it. To open a new job dialog box, right-click the Jobs folder and select New Job.

Jobs

The Job dialog box has three tabs: General, Job Steps, and Store Roles (Figure 54).

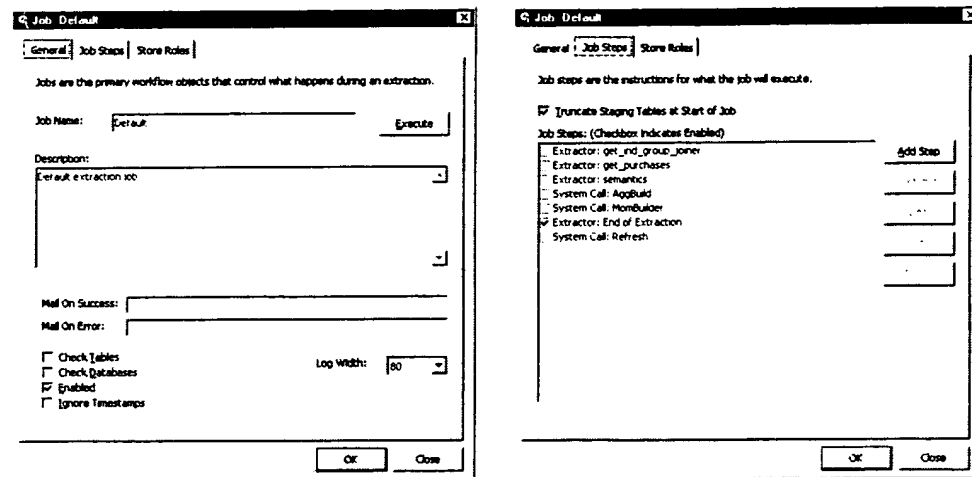


Figure 54: Job Dialog Box: General and Job Steps Tabs

The General tab provides options you can set that control the following aspects of a job:

- For a job other than the default one, assign a name and description.
- Enable a job. EpiChannel executes enabled jobs only.

New jobs are initially enabled. You may, however, disable a completely functional job in some circumstances to accommodate system changes. For example, if a database is in the process of being moved or repaired, jobs that populate that database could be disabled as a protection against accidental execution.

- Select the width of the log, either 80 or 132 columns. Select 80 for VGA monitors and 132 for SVGA monitors.
- Assign addresses for e-mail notification of the job's success or failure. See "Configuring E-Mail" on page 196.

- Request that before executing the job, EpiChannel check that all databases referenced by any job databases and all EpiMart tables are available. Unless you are sure that this is true, select this option.
- Request that all timestamps be ignored during job execution (Ignore Timestamp).

For example, by ignoring timestamps, you could retrieve all rows, without any date filters (based on EpiChannel's special filtering used for incremental extractions; see the Filters tab of the Extractor dialog box (Figure 50, on page 181)).

- Turn off the truncation of staging tables. (Use this only in development mode to restart a job without truncating any tables.)
- Execute a single job.

Click the Execute button in the Job dialog box (Figure 54) to execute this job. You can use the Execute Job dialog box (Figure 55) to do the following:

- Run the job as a trial run.
- Normally, the proper instance name is already selected. You can enter a new one by clicking the New button.
- Select a debug level that corresponds to the EpiChannel (**extract**) verbosity levels (although you cannot specify row numbers). This debug level takes effect before the first SQL statement.
- Indicate the total number of rows to be transferred in a pull/push operation: all rows or the first N rows. This value is reset with each extraction statement.

Warning: Truncation is enabled for this job.

Adding Extractors and System Calls as Job Steps

Using the Execute button is equivalent to invoking the **extract** command with this job as the job option. The Execute button, however, also supplies the database name, password, and so forth, if necessary (no instance is supplied) so that **extract** does not depend upon the Registry entries.

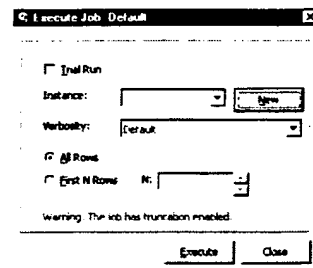


Figure 55: Execute Job Dialog Box: Selecting Debug Level

You can use the Store Roles tab to set up the data stores for the job (by default *EpiMart* and *JobFileLog*) and indicate their roles; for example, *working directory* or *log*. See “EpiChannel Output” on page 80. A *working directory* is required.

You can use the Job Steps tab to do the following:

- Enable job steps by selecting them in the check box to the left of their names. Only enabled job steps are run.
- Change the order of job steps, using the Up, Down, and Remove buttons.
- Update the job definitions in metadata when you modify the job steps. (Click Edit to open another dialog box in which you can update a job step.)

Adding Extractors and System Calls as Job Steps

You will also use the Job Steps dialog box to add one of your defined extractors as a job step and to create system calls. Click the Add Step button, which displays the dialog box shown in Figure 56.

Adding Extractors and System Calls as Job Steps

To add an extractor:

Step 1: Select Extractor.

Step 2: Select the extractor from the drop-down list, and click OK.

Click New to display the Extractor dialog box (Figure 50) in which you can create a new extractor.

Step 3: The extractor now appears in the Job Steps list.

As mentioned, sites with more complex databases may require multi-stages and additional commands, such as lookup tables, aggregation splits, gathering data into ranges (binning), and duplicate detection. For this purpose, you may use system calls, which are executed during a job as if invoked from the console's DOS command line.

To create a system call:

Step 1: Select System Call.

Step 2: Enter the name of the system call, the action to be taken upon error (abort, ignore, or print), the command line, and an optional description.

Step 3: Click OK to add the system call as a step.

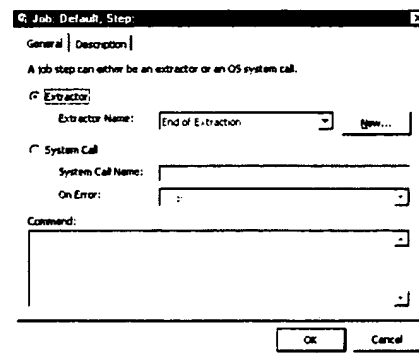


Figure 56: Add Job Step Dialog Box

External Tables

External Tables

Extraction statements are sometimes directed to load external tables that serve as intermediary tables for multi-staged extraction. Epiphany supplies one external table called *last_extract_date*. Use of this table is recommended, but optional. See “External Tables” on page 50.

To define external tables and their columns:

- Step 1:** Choose New External Table from the Extraction menu.
- Step 2:** Enter the name and any description for the table in the External Tables dialog box (see Figure 57).
- Step 3:** By default, the data in the external tables is deleted (truncated) after extraction. In most cases, you will accept this default.

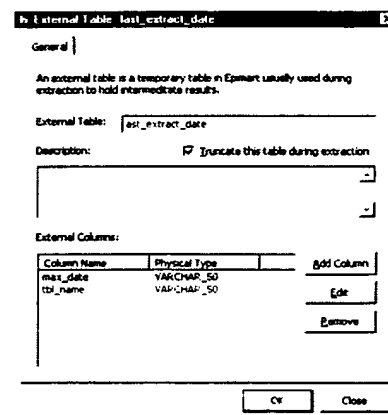


Figure 57: External Tables Dialog Box

To add a column:

- Step 1:** Click Add Column and enter the name in the External Column dialog box.

Step 2: Select the physical type. See Appendix D, “Physical Type Values” for descriptions of these physical types.

Add as many columns as necessary. After you click OK in this dialog box, the columns are added to the Column Name list in the External Tables dialog box.

Momentum Extraction

To run the Momentum application, use the built-in MomentumBuilder job (see Figure 58) in which MomentumBuilder is a system call job step. The order of the extraction is as follows:

Step 1: SQL/Semantics

Step 2: AggBuilder

Step 3: MomentumBuilder

Step 4: End of Extraction

For information about running MomentumBuilder as a standalone executable and to determine if MomentumBuilder extracted the correct data, see “MomentumBuilder” on page 60.

Configuring E-Mail

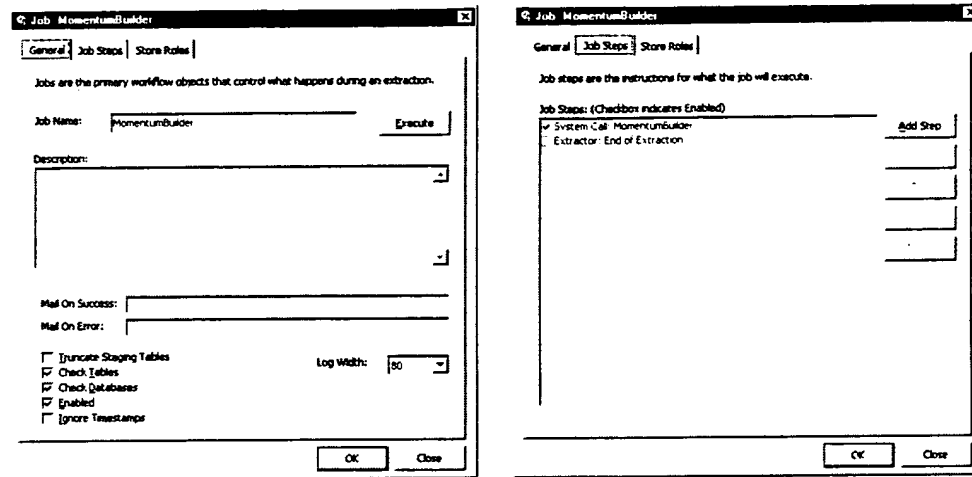


Figure 58: Momentum Job Dialog Box

Configuring E-Mail

To have the system automatically send e-mail notification of the outcome of a job (either successful completion or job failure), EpiCenter Manager needs to know your e-mail Profile name and password. These must match the information in the Mail and Fax Control Panel. Windows uses this Control Panel to define the profile of mail accounts, where each profile could tie users to different mail servers. The operating system then makes mail to this server possible via the SMAPI program interface used by EpiChannel.

To determine your mail profile name, follow these steps:

Step 1: Open the Mail and Fax Control Panel (from the Start menu, choose Settings/Control Panel and double-click Mail and Fax).

You may be given a list of profiles or may be placed into the only profile available. Either remember your profile name or click Show Profiles to display the name of the profile that was automatically opened.

Step 2: Choose Configuration from the EpiCenter menu.

The list of EpiCenter configuration variables includes the Mail Password and Mail Profile Name.

Step 3: Click Mail Profile Name and enter the name of the profile as shown in the Mail and Fax Control Panel in the Value textbox. Click Update.

Step 4: Click Mail Password and enter the password. Click Update.

The mail password depends upon which mail server is used and follows the rules of this server. Often, the password will be the same as the Windows NT user password.

Note: Because this password may be visible to others in mail logs or in metadata exports, do not use a password of any importance. Create a new mail user on your mail server exclusively for EpiChannel if revealing this password presents a problem.

Verifying that E-mail Notification Works

To test if e-mail is configured correctly, follow these steps:

Step 1: Open a Job dialog box for a job that simply runs and exits. In the General tab, enter the addresses for mail on success and mail on failure.

Step 2: Run the job.

If you receive e-mail notification, you have set up EpiChannel e-mail correctly. If not, repeat the steps for configuring e-mail given above and run another job. Please contact Epiphany Customer Support if there is still a problem.

Configuring Outlook Exchange for EpiChannel E-mail Notification

Follow these steps to configure Microsoft Outlook Exchange 98 for EpiChannel e-mail notification:

Step 1: Install Microsoft Outlook Exchange 98 according to the instructions on the screen. When prompted to select which kind of installation, select Corporate/Workgroup. Reboot your computer.

Truncating Tables

Step 2: Start Outlook Exchange, and choose to configure an Internet Mail account.

Step 3: When prompted, enter a Profile Name for this account.

Note: The Mail Profile in the EpiCenter Manager Configuration dialog box is the default e-mail address for the EpiCenter Manager user.

Step 4: Set up the account with an appropriate e-mail account name, user name, and organization.

Step 5: For the e-mail and reply address, enter a bogus address, such as *foo@bar.com*.

Step 6: In the Servers tab, set both the incoming and outgoing mail servers to the name of the mail server that can route mail outside the company's firewall. (This name is not case-sensitive.) In the Incoming Server section, enter any account name and password. (Because EpiChannel never uses Outlook Exchange to check incoming mail, these can be any values.) Accept the default values for the other tabs.

Step 7: Exit Outlook Exchange.

Note: If the mail server is installed on a UNIX platform, you can test it by logging on with a shell account and sending mail to yourself and Epiphany from the UNIX mail program.

Truncating Tables

By default, all staging tables and external tables are truncated prior to an extraction. You can set truncation on or off via individual fact, dimension, and external table dialog boxes, or use one dialog box to define all tables in a constellation. They both refer to the same fields in the database.

To define the set of tables for truncation:

Step 1: Choose Edit Truncation from the Extraction menu, or click the Edit Truncation toolbar icon in the toolbar menu.

Step 2: In the Truncate Tables before Extraction dialog box (Figure 59), select tables to be truncated.

All of your tables are listed in columns by type (Dimensions, Facts, and External tables). Check each one in the list that you want truncated. (Clicking the All button selects all of the tables in a column; clicking None de-selects all of them.)

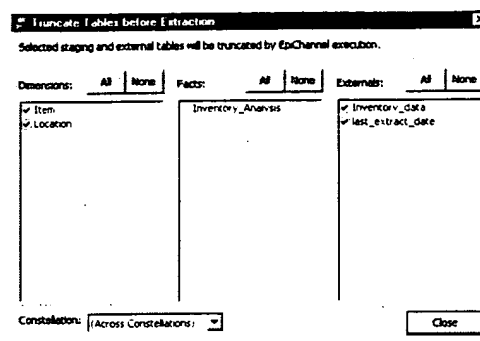


Figure 59: Truncate Tables before Extraction Dialog Box

Purging EpiMart Tables

You can remove tables from the EpiMart that are no longer needed. For example, you may have unused tables to delete or, as a result of metadata changes, have decreased the number of aggregates. These higher numbered aggregate tables remain in effect until you purge the EpiMart tables.

You should first try a trial run.

To purge database tables:

Step 1: Choose Purge EpiMart Tables from the EpiCenter menu.

Step 2: Select Trial Run in the Purge tab, and click Go.

Security

After the run, click the Results tab to see which tables will be deleted. If these results are acceptable, return to the Purge tab, de-select Trial Run, and click Go to delete these tables from your EpiMart.

SECURITY

The Epiphany system provides two areas of security:

- *Authentication*, or a user's ability to log on to the system. Authentication is determined by the Windows NT operating system.
- *Access rights*, or the permissions a user has after logging on. Access rights include the user's ability—
 - to open a ticksheet.
 - to save queries for those ticksheets. Saved queries (called *reports*) are lists of the options that a user selected on a ticksheet (to generate a report).
 - to access data based on the values of dimensional attributes; that is, to restrict access rights to the ticksheet to certain dimensional attribute values. You can use this kind of security to allow some users to see data for one region but not another.

The Epiphany system can also use Windows NT groups to administer access rights.

You will use the Security folder in the EpiCenter Manager directory tree to add groups and users to the system. The two work in tandem: groups must have users, and users belong to groups. When setting up a new Epiphany system, first set up groups and then add members after you have set up users. (Windows NT groups members can simply be imported.)

The Security folder also contains the Report Gallery, which enables the administrator to organize all saved reports in folders for groups and users. See "Report Gallery" on page 209 for more information.

Right-clicking the Security folder display a pop-up menu with commands for New Group, New User, Report Gallery, Import NT Group, Import NT User, and Export.

Setting Up Groups

Members of a group share similar access rights, or permissions. In general, you should attach these to groups instead of users to simplify maintenance. Access rights set for a user take precedence over access rights set for groups to which the user belongs.

To define a group, right-click the Group folder and select New Group from the pop-up menu. (This menu also has commands for exporting all groups, importing NT Groups, and refreshing the group folder list.) Complete the information requested in the Group Definition dialog box tabs (General, Membership, Ticketsheet Access, Column Access, and Description) as described in this section. (Use the Description tab to document the group for your own reference.)

Step 1: In the General tab (Figure 60), enter the group's name.

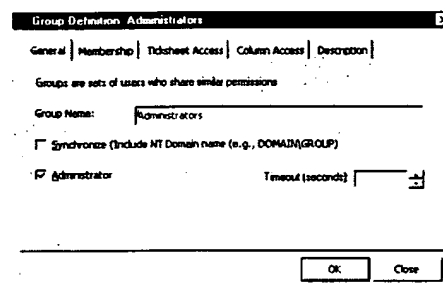


Figure 60: Group Definition Dialog Box: General Tab

Step 2: Select the Synchronize option to add new users to the group in an “autopilot” fashion via Windows NT synchronized groups. You need to precede Windows NT synchronized group names with their Windows NT domain name prefix (and matching name).

When the synchronize option is set, each time a user logs in, the Windows NT security API is accessed for a list of group names to which that user belongs. If the user is a member of an NT group for which there is no membership record in EpiCenter Manager, then a new record is

Setting Up Groups

created. Conversely, if the user is no longer a member of an NT group, but there is such a membership record in EpiCenter, then the latter record is removed.

- Step 3:** Select Administrator if members of this group have administrative rights. An Administrator can see all ticksheets and reports in the system, but cannot modify special folders, such as the Public folder. (See “Report Gallery” on page 209 for a discussion of Public folder.)
- Step 4:** Enter the seconds for the maximum time that a query will run for members of this group. As an administrator, you may set a time-out limit to ensure that someone does not monopolize database engine resources.
- Step 5:** You can use the Membership tab (Figure 61) to add already created users to the group. Users may be members of multiple groups. Click Add User.

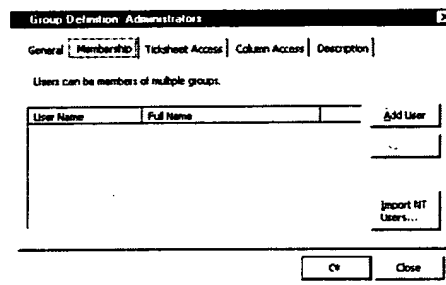


Figure 61: Group Definition Dialog Box: Membership Tab

- Step 6:** Select the user's name from the Choose User dialog box and click OK, or click New, which displays the User dialog box.
- Step 7:** To import NT users into the group, click Import NT Users and enter the domain in the dialog box.
- Step 8:** Select one or more members from the Choose NT Users dialog box.

Step 9: You can use the Ticksheet Access tab to assign all members of the group access to certain ticksheets. Before Epiphany users can open any ticksheet, they must be granted access to it. Users have the ability to see and use all ticksheets granted access to their groups by the administrator. Click Add Ticksheet and select ticksheets from the Choose Ticksheet dialog box. The dataset that the ticksheet belongs to and ticksheet type are displayed in the tab.

Step 10: When a user clicks the Create Report button in a ticksheet to generate a report, some results may be filtered automatically so that the report's output is restricted.

You can use the Column Access tab (Figure 62), which is the same for users and groups, to restrict access rights for the ticksheet to certain attributes (dimension columns).

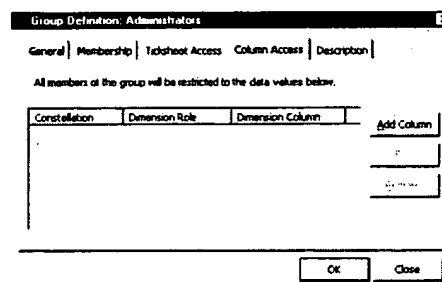


Figure 62: Group Definition Dialog Box: Column Access

Setting Up Groups

Click the Add Column button, which displays the Dimension Column Access Group Definition dialog box (Figure 63).

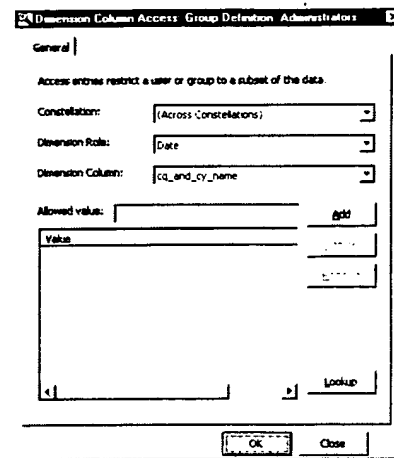


Figure 63: Dimension Column Access Group Definition Dialog Box

Step 11: Select the Constellation to which the dimension column belongs from the drop-down list. You may also apply this restriction across all constellations (for example, apply the date dimension).

Each ticksheet is associated with a constellation. When a user runs a report, only those dimension column restrictions that apply to that constellation are applied. Thus if an EpiCenter has two constellations, one for Sales and one for Expense, there will be no filtering on Expense dimensions when a user works with a Sales ticksheet (unless Across Constellations has been selected).

Step 12: Select the Dimension Role for this dimension column from the drop-down list.

Step 13: To display all of the values in the dimension column, click Lookup, which directly accesses the EpiMart data. Select a value or values from the listing and click OK to add it to the value list.

Setting Up Users

The values entered in this list should correspond to actual database values in the base dimension table to which that dimension column corresponds. For example, selecting *Date.fy_name* as the field with the values 1997 and 1998 causes all reports to be filtered with these values.

Step 14: If you know the exact value of the dimension column whose values will be accessible to the group, type it in the Allowed values textbox, and click Add to place it in the Value listing below. Repeat this step to add additional values.

Step 15: Click OK to return to the Group Definition tab.

After a Group has been defined, it becomes a subfolder of the Group folder. You can right-click a group folder and use the pop-up menu to set up a new group, or to edit, delete, or duplicate the group. A duplicate group is identical except for its group name.

Setting Up Users

All authorized users of the Epiphany system will appear as icons in the Users subfolder of the Security folder. By default, new users have these permissions:

- no access to any ticksheets.
- no group membership.
- inability to save any reports.
- access to all data in all dimensions—there are no dimensional attribute restrictions.

The access rights set for a user have precedence over access rights set for groups to which the user belongs.

To set up a new user:

Step 1: Right-click the Security folder and select New User. The User dialog box (Figure 64) that displays has tabs for General, Membership, Ticksheet Access, and Column Access.

Setting Up Users

Step 2: You can use the General tab to enter the person's username and first and last names.

The first and last names are used for display purposes only. If they are missing, then the username is displayed.

For Windows NT authentication, enter users with their Windows NT domain prefix to distinguish among users with the same name but who are in different domains.

Step 3: Enter the seconds for the maximum time that a query will run for this user. As an administrator, you may set a time-out limit to ensure that a user does not monopolize database engine resources.

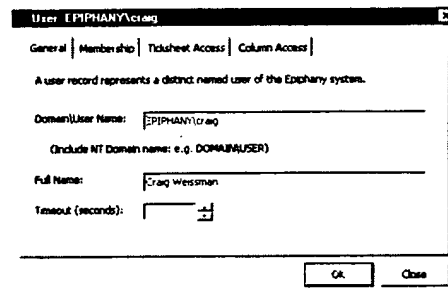


Figure 64: User Dialog Box: General Tab

Step 4: After groups have been created as described in "Setting Up Groups" on page 201, you will use the Membership tab to assign group memberships to the user.

Click Add Group and select one or more groups from the Choose Groups dialog box. (Hold down the Shift key to select more than one.) A user can be a member of multiple groups, but for security reasons needs to be a member of a single group (called the primary group) when running queries. After assigning group memberships, select the main group in the list and then select the Primary option in the tab.

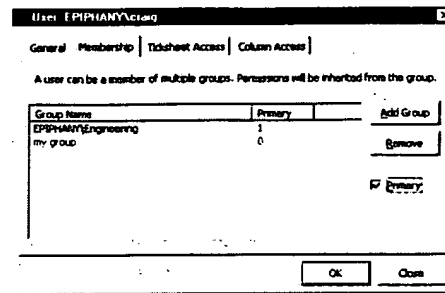


Figure 65: User Dialog Box: Membership Tab

- Step 5:** Before Epiphany users can open any ticksheet, they must be granted access to it. Users have the ability to see and use all ticksheets to which they have access, and to all ticksheets granted access to their groups. In the Ticksheet Access tab, click Add Ticksheet and select the ticksheets in the list that the user may access. Hold down the Shift key to make multiple selections.
- Step 6:** You can use the Column Access tab (Figure 66) to restrict access rights for the ticksheet to certain attributes (dimension columns). Click Add Column, which opens the Add Column dialog box.

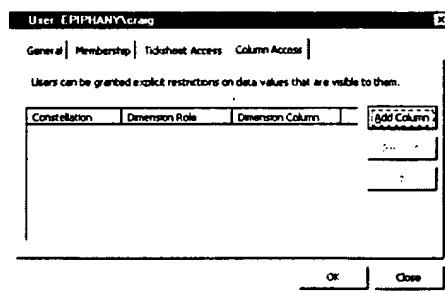


Figure 66: User Dialog Box: Ticksheet Column Access Tab

Setting Up Users

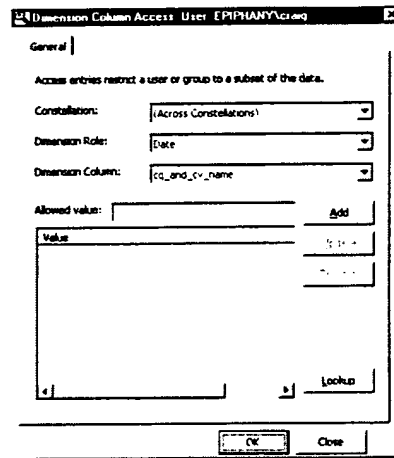


Figure 67: Dimension Column Access Dialog Box

- Step 7:** Select the Constellation to which the dimension column belongs from the drop-down list. You may also apply this restriction across constellations. Each ticksheet is associated with a constellation. When a user runs a report, only those dimension column restrictions that apply to that constellation are applied. Thus if an EpiCenter has two constellations, one for Sales and one for Expense, there will be no filtering on Expense dimensions when a user works with a Sales ticksheet (unless you have selected Across Constellations).
- Step 8:** Select the Dimension Role for this dimension column from the drop-down list.
- Step 9:** To display all of the values in the dimension column, click Lookup, which directly accesses the EpiMart data. Select a value or values from the listing and click OK to add it to the value list.

Report Gallery

The values entered in this list should correspond to actual database values in the base dimension table to which that dimension column corresponds. For example, selecting *Date.fy_name* as the field with the values 1997 and 1998 causes all reports to be filtered with these values.

Step 10: If you know the exact value of the dimension column whose values will be accessible to the group, type it in the Allowed values textbox, and click Add to place it in the Value listing below. Repeat this step to add additional values.

Step 11: Click OK to return to the User dialog box.

Report Gallery

Note: The Report Gallery in EpiCenter Manager is for administrative use only. Many of the features of this Report Gallery can also be accomplished by end users using the Report Gallery in the Web interface.

You can use EpiCenter Manager to browse the saved reports for a ticksheet, as well as view who has access to those reports. All of the saved reports for the Epiphany system are organized in folders in the Report Gallery. There are folders for users and groups, and a Public folder. As mentioned, whenever a new ticksheet is created, the administrator should save one report in the Default subfolder in the Report Gallery's Public folder (see "Report Gallery" on page 209) and make it the default.

To display the Report Gallery, right-click the Security folder and select Report Gallery from the pop-up menu. The Report Gallery (see Figure 68) is displayed. The top panel shows the folders organized in a tree hierarchy, and the lower panel lists the reports for a selected folder by report name, report type, ticksheet name, date last modified, and the person who modified it.

Report Gallery

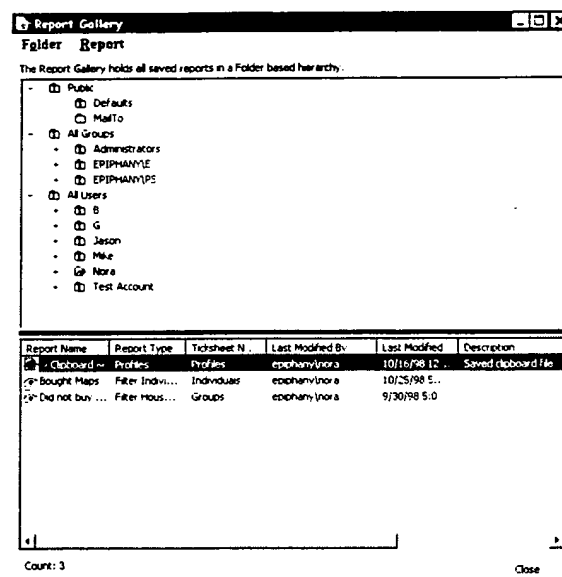


Figure 68: Report Gallery in EpiCenter Manager

Each group and user has a default report folder called Defaults, which is located in the user's or group's folder. There is also a Public Defaults folder.

There is a default report for each ticksheet at the user and group levels. A default report is the set of ticksheet settings that a user sees when he or she first opens that ticksheet. In general, the most specific default report is used. Thus if a user-level default report exists, that report is opened in the user's Web browser. If no user report exists but a group default report exists for the ticksheet, the group default report is displayed. The default report in the Public folder is available if there is no specific user or group default report.

When a user saves a report, it is saved in whatever folder the user is currently in. This can be the user's default folder, a folder for which the user has write access, or the Public folder.

The Report Gallery's main menu has menus for Folder and Report, which are described below.

Folder Menu

Note: Right-clicking in the top panel of the Report Gallery displays a pop-up menu with the Folder commands.

You can use the Folder menu to—

- create a folder for saved reports. Choose New from the Folder menu, select the folder or subfolder, and enter the new folder name in the dialog box.
- delete a selected folder.
- display the folder's properties. (You can also double-click the leaf folders to open the Folder dialog box.)

The Folder dialog box has two tabs: General and Permissions. The General tab shows the folder's name, path, date last modified, and any description.

The Permissions tab allows you to set controls for who can view and alter the folder. See "Setting Permissions" on page 212 for more information.

- find a report in one of the folders.

Enter the file name in the Find Report dialog box. You can refine your selection to report types and date modification ranges. Pressing the F3 key locates the next report that meets this criteria.

- Move folders by dragging them.

Report Menu

Note: Selecting a report file in the lower panel displays a pop-up menu with the Report commands.

You can use the Report menu to copy and move reports and folders. (This functionality is not available through the Web interface Report Gallery.)

To copy a report:

Select it, and choose Copy from the Report menu. Select the folder in which you would like to place the copy of the report, and select Paste.

You can also cut and paste a report, or explicitly delete it.

Report Gallery

Double-clicking a report opens the Report dialog box (Figure 69).

- The General tab shows the report's name, folder path, report type, ticksheet name, date last modified, and any description.
- The Permissions tab allows you to set controls for who can view and alter the report. See “Setting Permissions” for more information.

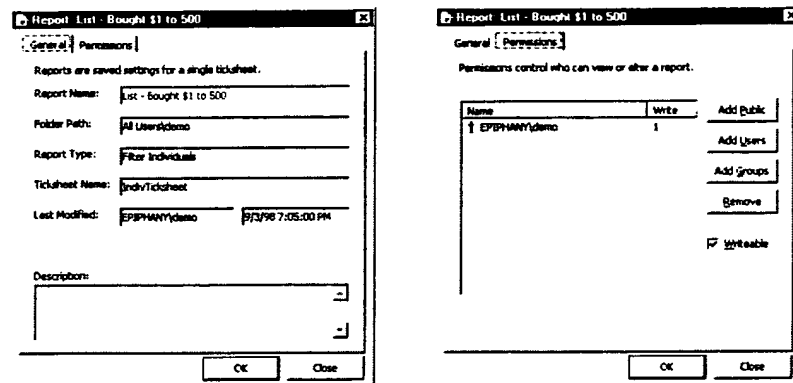


Figure 69: Report Dialog Box

Setting Permissions

You can use the Permissions tab of the Folder and Report dialog boxes (see Figure 69) to assign users and groups permission to view a folder or report. If you select Writable in the Report Permissions tab before clicking the Add button, then the users or groups you add may alter the report. Selecting Writable in the Folder's Permissions tab gives write access to any report in the folder for the user or group (although a user cannot create a new report in a folder unless he or she has write access to that folder).

- To allow all users the ability to view this folder or report, click Add Public.
- To give users access to folders/reports, click Add Users and select users from the Choose dialog box.
- To give groups access to folders/reports, click Add Groups and select groups from the dialog box.

SHARED INTERFACE

The Shared Interface folder has the subfolders Datasets and Glossary Entries, which are user-interface features shared by the ticksheets.

As part of building a ticksheet, you added Glossary entries to describe ticksheet terms to users. These terms are hyperlinked to a glossary page. When a user clicks a link, a glossary page displays which defines it. See “Setting Up a Glossary Entry” on page 134 for instructions.

A dataset is a logical grouping of similar ticksheets within a constellation. Instructions for setting up datasets follow.

Setting Up Datasets

You can group related ticksheets by dataset. In the browser window, these group names, or labels, appear in the main menu (left margin of the ticksheet) under the Dataset heading. For example, one dataset may contain Executive Summary ticksheets and another Shipment-related ticksheets.

A dataset may contain one and one only of each ticksheet type. For example, the Executive Summary dataset can have only one Clarity Table & Charts ticksheet, one Relevance Best & Worst, one Relevance Influence, one Relevance Lifecycles, and so forth.

You can use the Dataset dialog box (Figure 70) to create and describe these logical data views. When you build a ticksheet, you will assign it to a dataset (see “Assigning the Ticksheet to a Dataset” on page 127). Right-click the Dataset folder and select New Dataset. Enter the dataset name and the label name (the name as it should appear on the ticksheet). Enter any help text that will be displayed to end users.

Note: Most top-level entries in the EpiMeta need a unique name so that they can be identified for export/import purposes. The dataset name is internal to EpiMeta; users see only the label.

Generating Schema

The Ticksheet tab lists the ticksheets in the dataset. Dataset ordering applies after you have created more than one dataset. To rearrange the order in which the Datasets appear to end users, right-click a dataset in the Shared Interface folder, and select Up or Down from the pop-up menu.

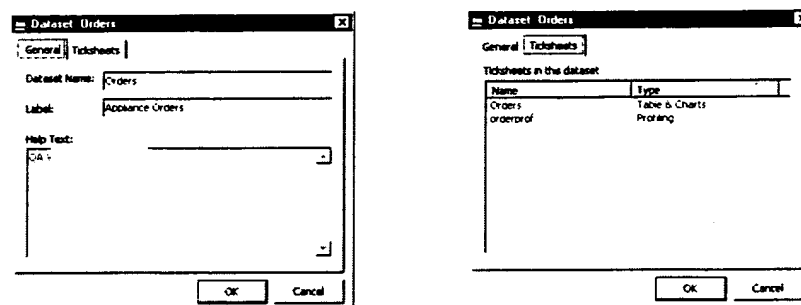


Figure 70: Dataset Dialog Box

GENERATING SCHEMA

After defining your metadata via EpiCenter Manager, you need to convert these definitions into actual tables (the EpiMart). You can use the Generate Schema dialog box to build the EpiMart tables, as well as to populate the physical date dimension table. The Date dimension table is a special base dimension table supplied by Epiphany for storing all attributes related to time. All fact tables in an EpiCenter receive a foreign key (called *date_key*) to this table.

Momentum users can use this dialog box to build the sampling table. The sampling table is a list of random numbers that efficiently produce statistically accurate samples. You can use the General tab of the Configuration dialog box to set the compression ratio for Momentum sampling (see “General Settings” on page 285).

To generate the schema for your EpiMart:

Step 1: Choose Generate Schema from the EpiCenter menu. The Generating EpiMart Schema dialog box has two tabs: Schema and Results.

Generating Schema

- Step 2:** In the upper panel of the Schema tab, select Trial Run to see what the results will be without making permanent changes to the files.
- Step 3:** By default, only tables that have changed since the last time you generated schema are rebuilt. Select Force Rebuild of all Tables only if you want to rebuild *all* of your tables.
- Step 4:** Momentum users have the option of selecting Build Sampling Table. A sampling table is a table that contains a statistical sample of the data.
- Step 5:** The date dimension table must be populated the first time you generate schema. Follow the instructions in “Populating the Date Dimension Table” below.
- Step 6:** Click Go (and watch the progress bar).
- Step 7:** After the trial run is finished, click the Results tab to view which tables were updated.
- Step 8:** If the results are acceptable, de-select Trial Run in the Schema tab, and click Go.

After the schema has been generated, the system records the current state of the metadata so that the next time the schema is updated, the current definitions can be compared with the new ones, and the appropriate tables can be created or altered as necessary.

Populating the Date Dimension Table

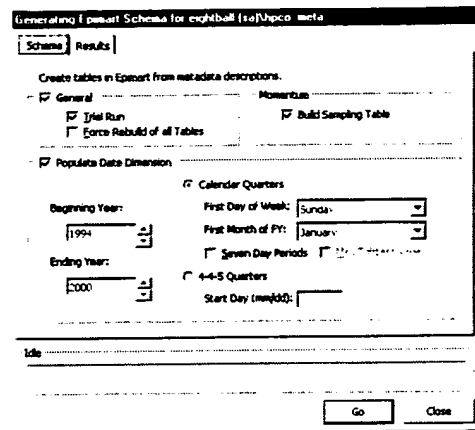


Figure 71: Generating EpiMart Schema Dialog Box

Populating the Date Dimension Table

To populate the date dimension table:

- Step 1:** Select Populate Date Dimension in the Schema tab of the Generating EpiMart Schema dialog box. (You can also choose Populate Date Dimension from the EpiCenter menu to populate the dates only; see "Populating the Date Dimension" on page 91.)
- Step 2:** Enter the values for the beginning and ending years of the EpiMart. The date range of the EpiMart should be at least as large as any dates that are found in the data you will be extracting. These values are defined in the Configuration dialog box (choose Configuration from the EpiCenter menu to open it).

For users to obtain the best results when forecasting trends using Relevance, you need to build the date dimension as far into the future as you plan to forecast. (Currently, the maximum prediction is three years

past the last date that has recorded data.) If the date dimension is not built out far enough, the user will receive columns with names such as *Dec 1999*, *Second Next*, *Third Next* instead of *Dec 1999*, *Dec 2000*, *Dec 2001*.

- Step 3:** Choose the appropriate quarter system: Calendar Quarters (three month divisions of the year) or Quarters 4-4-5. *4-4-5 Quarters* represents the 13-week-per-quarter calendar in which the months in the quarter are defined as consisting of 4 weeks, 4 weeks, and 5 weeks.
- Step 4:** *For Calendar Quarters*, enter the first day of the week and the first month of the fiscal year. Select Seven Day Periods to ensure that all fields that count up in number (such as *week_number_fq*) begin with seven day periods (that is, 11111112222223333333...13 or 14), whereas all fields that count down in number (such as *week_number_til_end_gq*) end with seven day periods (that is 14 or 13...33333332222221111111). The alternative is that weeks may overlap quarter boundaries incompletely.
- Step 5:** *For Calendar Quarters*, select Max Thirteen Week to prevent a “14th” week or “53rd” year; the 13th or 52nd weeks are simply extended as needed.
- Step 6:** *For 4-4-5 Quarters*, enter the start date of the first quarter.

Appendix , “Date Dimension Fields” describes the date dimension fields used by the Epiphany system.

EXPORTING/IMPORTING METADATA

Epiphany provides a metadata export/import utility for moving metadata between EpiCenters and for backing up the definition of an EpiCenter. To use this tool properly, you need to understand the metadata concepts (see “Metadata Overview” on page 323).

Exporting/Importing Metadata

The export operation produces a Microsoft Access database that contains a representation of the metadata that was chosen for export. Upon import, this representation is converted back into valid Epiphany metadata in the target EpiMeta. Using EpiCenter Manager, you have granular control over which metadata objects get exported during each operation. For example, you can use the same constellation definitions, extractors, and ticksheets in several EpiCenters. Also, during import, you control whether or not to overwrite existing metadata that conflicts with what is in the import file.

Reasons for exporting files other than publishing them for import are to save files for document control (source safe) and to send files as an e-mail attachment; for example, you might e-mail an exported file to Technical Support for analysis.

You can export individual metadata objects by right-clicking folders in the EpiCenter Manager directory tree and selecting the Export command from the pop-up menu. This option is available only for exportable metadata objects.

To display a dialog box in which you can export all or any subset of metadata files:

Step 1: Right-click the EpiCenter icon and select Export\Export All from the pop-up menu, which displays the Exporting Metadata dialog box (Figure 72).

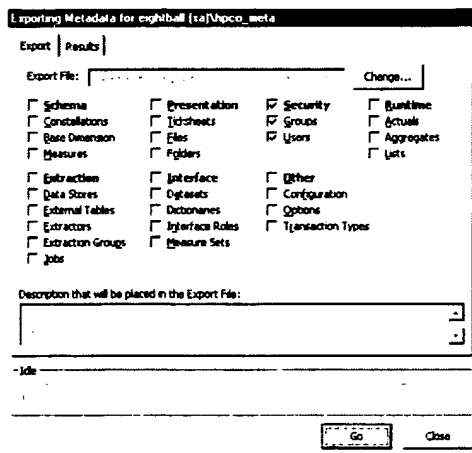


Figure 72: Exporting Metadata Dialog Box

Exporting/Importing Metadata

- Step 2:** In the Export tab of the Exporting Metadata dialog box select a heading, such as Schema to select all of the Schema topics automatically, or select individual subtopics.
- Step 3:** Enter any description that will be placed in the export file. This description can be read in the Importing Metadata dialog box when the user selects this file for import.
- Step 4:** Click Go.

To import individual metadata files in EpiCenter Manager, right-click a folder and select Import from the pop-up menu. This option is available only for metadata files that can be imported.

You can produce a new EpiCenter by importing all of the metadata files from an existing EpiCenter:

To import all of the metadata:

- Step 1:** Right-click the EpiCenter icon and select Import Metadata from the pop-up menu.
- Step 2:** The default export file is **export.mdb** in your **ConfFiles** directory. Click Change to select another file.
- Step 3:** Select Always Replace Existing Data if you want the new data to replace existing entries of the same name.
- Step 4:** Select Continue after Errors so that should an error occur, you will receive a partial report.
- Step 5:** Click Go.

Exporting/Importing Metadata

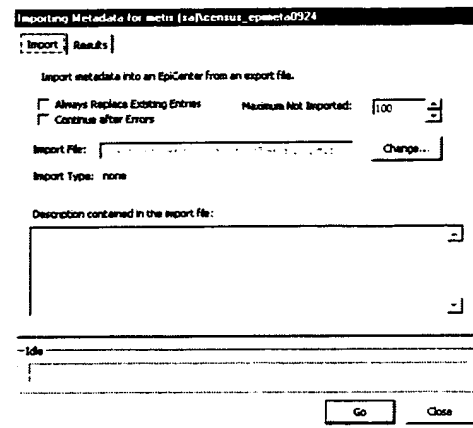


Figure 73: Importing Metadata Dialog Box

Note the following:

- Re-importing objects does not delete references to those objects. For example, a measure definition can be re-imported without deletion of the measure mappings in ticksheets that point to that measure.
- To speed up import and export, the entire Microsoft Access database is either read from or written to at once.
- All import operations are performed inside of a single database transaction. Any error that occurs during import can be rolled back completely.

See Appendix G, “Export/Import of Metadata” for more information.

TOGGLE A AND B TABLES

To switch to the A set of tables from the B set, or vice versa, choose EpiCenter\Toggle A & B Tables, or click the Toggle A & B Tables toolbar button.

WEB BUILDER

As a security measure, the functions necessary for building ticksheets are available through a separate program called Web Builder. Access to only the Web Builder program is appropriate for someone who is authorized to create ticksheets for an already initialized EpiCenter, but does not need access to all of the features of EpiCenter Manager.

During installation this program executable (**WebBuilder.exe**) is placed in the Epiphany directory by default (**C:\Program Files\Epiphany**) and can be selected from the Start menu: Programs\Epiphany*instance_name*\EpiCenter Manager (User Interface Only). After you run the executable, the EpiCenter Enterprise Manager window is displayed. At the top level is the Servers icon. Follow these steps to register your server:

Step 1: Choose Register from the Server menu. The Server Properties dialog box (Figure 9, on page 88) is displayed in the window.

Security Manager

Step 2: Select the Server Type from the drop-down list and enter the database server's name, and the username and password for this server.

Your database server machine icon and the EpiCenters folder (or directory) appear in the hierarchical tree structure. This server icon represents the server where the EpiMart and EpiMeta databases for this EpiCenter reside.

Step 3: Select the EpiCenter you plan to work with from the Choose EpiCenter dialog box.

In the EpiCenter that is displayed, only two folders are shown: the Constellations folder with Measures and Ticksheets only, and the Shared Interface folder, which shows only Glossary Entries and Datasets.

See "Configuring a Clarity or Relevance Ticksheet" on page 126 for instructions.

SECURITY MANAGER

As a security measure, the security functions of EpiCenter Manager are packaged as a separate program. Security Manager is available for use by those responsible for setting up access rights for users and groups and permissions for ticksheets, but who do not need access to all of the other features of EpiCenter Manager.

During installation this program executable (**SecMgr.exe**) is placed in the Epiphany directory by default (**C:\Program Files\Epiphany**) and can be selected from the Start menu: **Programs\Epiphany\instance_name\EpiCenter Manager (Security Only)**. After you run the executable, the EpiCenter Enterprise Manager window is displayed. At the top level is the Servers icon. Follow these steps to register your server:

Step 1: Choose Register from the Server menu. The Server Properties dialog box (Figure 9, on page 88) is displayed in the window.

Step 2: Select the Server Type from the drop-down list and enter the database server's name, and the username and password for this server.

Running the Scrutiny Debugging Tool

Your database server machine icon and the EpiCenters folder (or directory) appear in the hierarchical tree structure. This server icon represents the server where the EpiMart and EpiMeta databases for this EpiCenter reside.

Step 3: Select the EpiCenter you plan to work with from the Choose EpiCenter dialog box.

The EpiCenter contains only the Security folder. See “Security” on page 200 for instructions.

RUNNING THE SCRUTINY DEBUGGING TOOL

Scrutiny is an interactive debugging tool within EpiCenter Manager. It performs a subset of checks available in the Epiphany Application Server without your having to examine log files, and attempts to fix any problems it finds.

Scrutiny ensures that your EpiCenter (both EpiMeta and EpiMart) is in a consistent and functioning state. It does this by running an extensive set of queries against your EpiMeta and EpiMart to ensure various rules are followed and multiple descriptions of items are consistent. For example, Scrutiny can examine the EpiMart tables for missing indexes, missing rows, bad referential integrity, and so forth. It can also catch Momentum constraints such as not allowing group transaction filters on an individual ticksheet. It will also check consistency constraints, such as the presence of UNKNOWN rows in EpiMart tables and well-formed references.

You can run Scrutiny periodically as a validation tool, or whenever you encounter any problems with metadata or EpiMart data, or if the Application Server fails to start. Scrutiny fixes and diagnoses many common (and less common) issues.

Running the Scrutiny Debugging Tool

Choose Run Scrutiny from the EpiCenter menu. In the Scrutiny dialog box (Figure 74), select whether you want to perform EpiMeta or EpiMart checks, or both. For EpiMart, you can select the A or B set of tables, or both. In general, Scrutiny runs quickly (a few minutes for large EpiCenters), especially when EpiMart is not selected.

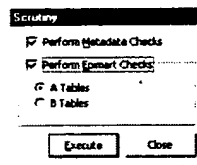


Figure 74: Scrutiny Debugging Tool

When Scrutiny runs, it displays a text screen of the checks that are running. If an error is detected, it describes the nature of the error and either how to fix it yourself (usually in EpiCenter Manager) or proposes a solution for you. If you would like it to execute its proposed solution, press y. After Scrutiny has executed, it asks if you want to re-run that section, to ensure that the fix was applied successfully.

Important: In some cases Scrutiny fixes are last resorts, and should be applied only if all else fails. These checks are identified as such, but be sure to read the descriptions carefully before applying any fix.

CHAPTER 4

EPIPHANY APPLICATION SERVER

The Epiphany Application Server is the component of the Epiphany Application Suite that processes all user requests and returns query data in HTML format. All Epiphany applications run on top of the Application Server. The Application Server has a multi-threaded design that allows several interactive connections to occur simultaneously.

In general, an Application Server interaction begins when the user points a browser at a Web URL address. The URL is first processed by the Web server (IIS), and then routed to the Epiphany proxy, **Epiphany.dll**. The proxy packages the request and sends it to the Application Server via a TCP/IP socket. The proxy waits for the Application Server to process the request and return a result. The proxy then takes the result and returns it to the user's browser via IIS. The proxy serves to separate the Application Server from the IIS process. This architecture allows several IIS machines to point at the same Application Server. The Application Server listens for requests, dispatches them to work or threads, then processes the requests and returns the results. Figure 75 shows the role of the Application Server in the Epiphany system.

An installation may run multiple Application Servers that are installed on one or separate machines. For example, a second Application Server might serve as a backup, or one Application Server might be used for development and another used for production.

Epiphany Application Server

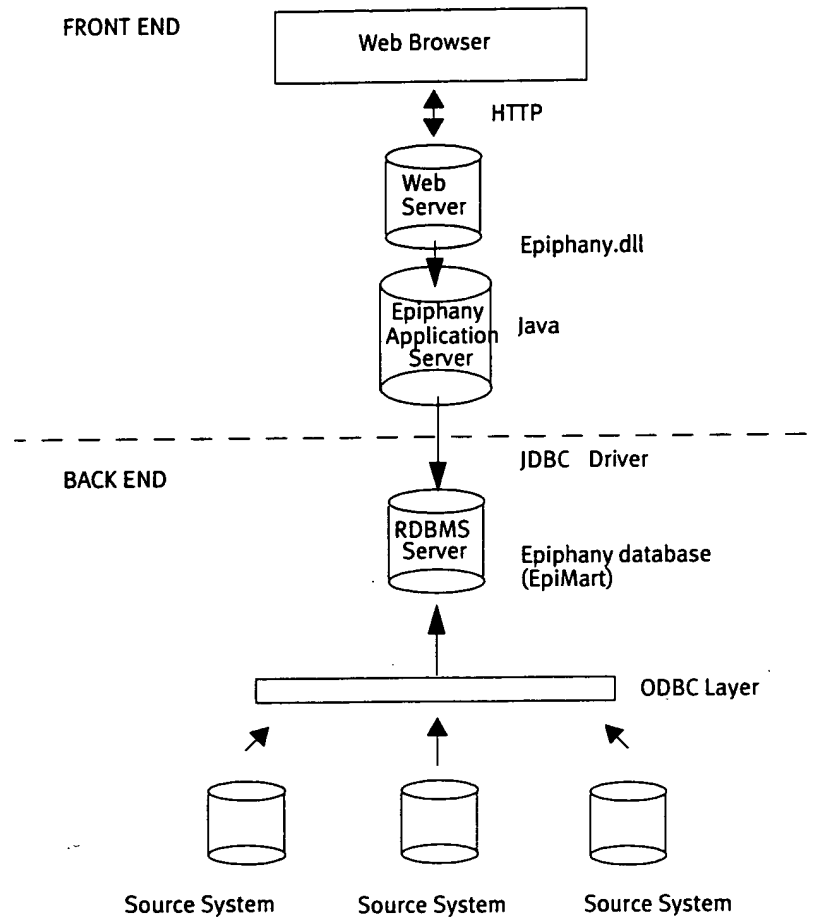


Figure 75: Epiphany System Diagram

The Application Server is a Java application that is normally run as an NT Service. Java is not used on the client (browser) side.

This chapter covers the following topics related to the Epiphany Application Server:

- Starting and stopping (see page 227)
- Command-line arguments (see page 232)
- Refreshing (see page 232)
- EpiAppService program (see page 237)
- Epiphany proxy (see page 243)
- Logging (see page 245)
- Security (see page 250)
- Administrative Groups (see page 256).

For instructions on how to troubleshoot the Application Server, see Appendix H, “Troubleshooting.” This appendix also describes the Application Server error messages.

Note: If the Application Server won’t start, you can run the Scrutiny debugging tool in EpiCenter Manager to diagnose the problem. See “Running the Scrutiny Debugging Tool” on page 223.

STARTING AND STOPPING THE SERVER

This section gives instructions for starting and stopping the Application Server when you run it as a service and when you run it as a console application.

Running as a Service

A standard Epiphany software installation (as described in the *Epiphany Installation Guide*) installs the Epiphany Application Server as an NT service. You can start and stop the service using the Windows *Control Panel\Services* menu. Follow these steps:

Step 1: From the Start menu, choose *Settings\Control Panel\Services*.

Determining if the Application Server Is Running

Step 2: Select the Epiphany Application Server. It is usually installed under the same name as the *instance_name* you specified during installation.

Step 3: Click Stop to stop the Application Server.

Step 4: Click Start to start it.

Determining if the Application Server Is Running

When you run the Application Server as a service, it may be difficult to tell whether it is running. To determine if the Application Server is running:

Step 1: Check if the service is running in the Control Panel.

Step 2: Check the NT Event log by going to the *Start\Programs\Administrative Tools (common)\Event Viewer* and opening the *Log\Application* menu.

If the Service was started and is running, there will be an entry with Source = EpiAppServer. The message reads Epiphany Appserver <instancename> message: The Service was started.

Step 3: Check for the existence of the Application Server's log file.

The Application Server creates a log file immediately upon initialization, which will be named using the following convention:

1999-02-08_14-48-15-32SRV.txt

where 1999 is the year, 02 is the month, and 08_14-48-15 is the day, hour, minute and second when the Application Server started running.

The log file resides in the logging directory, which is specified in the Windows Registry under the key:

HKEY_LOCAL_MACHINE\Software\Epiphany\Instances\instance_name\SystemLogDir.

Normally, this log file is located under the directory:

\instance_name\web\WWWROOT\logfiles.

If you have started the Application Server and it has been initialized, this log file resembles the one shown in Figure 75.

Determining if the Application Server Is Running

```
=== EPIPHANY APPLICATION SERVER =====
=== Version 3.4.0.1285
==Tue Jan 20 11:05:57 PDT 1999
Instance: momentum_proust3
Adding session: __ global __|unspoofable
[EpiCenter] Initializing instance momentum_proust3
  EpiMeta : lightfoot/hxco33_epimeta0924 {user=sa}
  EpiMart : lightfoot/hxco33_epimart {user=sa}
  Momentum: =====Starting momentum special dimensions load
  Momentum: Got a household dim base of ICN
  Momentum: Got a individual dim base of Site
  Momentum: Finished Loading int map table list
  Momentum: =====Finished momentum special dimensions load
  [TimeNav] Initializing instance momentum_proust3
  [TimeNav] Initialized instance momentum_proust3 in 4426 ms
  Momentum: =====Starting momentum metadata load
  Momentum: Got a measure label of Dollar Amount
  Momentum: Got a measure label of Average Sale Price Shipment
  Momentum: Scanning non-momentum ticksheets for 11 ticksheets.
  Momentum: =====Finished momentum metadata load
[EpiCenter] Closing connections.
[EpiCenter] Initialized instance momentum_proust3 in 16373 ms
AggNav initializing instance momentum_proust3
AggNav initialized instance momentum_proust3 in 1061 ms
Security manager initializing instance momentum_proust3
Security manager initialized instance momentum_proust3 in 630 ms
Storage manager initializing instance momentum_proust3
Storage manager initialized instance momentum_proust3 in 341 ms
[om] Beginning init in C:\PROGRAM FILES\EPIPHANY\MOMENTUM_PROUST\WEB\TEMPLATES\
[om] Processed 86 templates.
Initializing encryption engine...
Server instance created!
Server Host: localhost
Server Port: 8088
Server Inst: momentum_proust3
**** Epiphany AppServer momentum_proust3 awaiting connections... ****
```

Figure 76: Sample Application Server Log File

Running as a Console Application

Running as a Console Application

Usually, you will start Epiphany Application Server from the Start menu: *Settings\Control Panel\Services* menu. During debugging and the initial setup, however, it may be expedient to start the Application Server from a console window because all of the logging information is copied to the console.

You can manually start the Application Server by using the **jre** program (in your local path) from the directory:

C:\Program Files\Epiphany\instance_name\classes.

Enter the following command line:

```
jre -mx64M -classpath .;.\connect;.\connect.jar;.\
\EpiAppServer.jar;C:\PROGRA-1\
JavaSoft\JRE\1.1\lib\rt.jar com.epiphany.server.Server
localhost:8081 @instancename
```

Notes:

If the **jre** program directory is not in your local path, you will have to pre-pend the pathname to the **jre** command. Normally, **jre** is installed in the

C:\Program Files\Javasoft\jre\1.1\bin directory. This command example also assumes that the Application Server is running on the local machine on port 8081 (localhost:8081). The **-mx64M** parameter allows the Application Server to allocate up to 64 megabytes of memory.

For security to function properly, the user who runs this command line must have certain NT user rights. Otherwise, no one will be able to log in (you will receive `EpiLoginException` violations).

When you manually start the Application Server using the **jre** command, a console window that echoes all of the logging information is displayed on your screen. The window must remain open while the Application Server runs. If you close this console window, the Application Server terminates. If you log out of your machine, the console window closes, and the Application Server terminates.

For Authentication to Work

For authentication to work, run the Application Server as a service under the Local System account, or under an administrator account that has special NT privileges. You will receive Error 1314 if privileges are not set. The special NT privileges are—

- Act as part of OS
- Increase quotas
- Replace a process level token

Note: You can use the pass through security module without setting these privileges. See “Authentication Modules” on page 252.

The Local System account already has these privileges, and almost always, installation will be set up to run the Application Server under the Local System account.

To run the Application Server from the command line, follow these steps:

Note: This procedure affects only the local machine, not the NT network.

Step 1: Log in as a user who has local machine administration access.

Step 2: Assign special NT privileges to a user account. To do so, start User Manager and enter the local machine name in the Select Domain dialog box.

Step 3: Choose Policies\User Rights from the menu.

Step 4: Click Show Advanced User Rights and assign the account of the currently logged-in user to have the privileges specified above.

Step 5: Reboot your machine for the privileges to take effect.

COMMAND-LINE ARGUMENTS

The Application Server's main routine is the Java program **jre**, which is located in the **com.epiphany.server.Server** class. The **jre** program has the following command-line arguments:

```
jre -mx64M -classpath classpath com.epiphany.server.Server  
      machinename:port number @instancename DEBUG=1
```

If the machine name and port number are not specified, then they are read from the Registry under the ***instance_name*** directory. See “The Application Server’s Registry Keys” on page 240 for more information. If the ***instance_name*** is not specified, then the server will read the following file to determine the default ***instance_name*** to use:

HKEY_LOCAL_MACHINE\Software\Epiphany\Instances

The **DEBUG=1** parameter will cause the Application Server to log extra information. This extra information includes the data about the “clean-up” routines that handle timed-out sessions and old command threads that have finished.

REFRESHING THE APPLICATION SERVER

There are two ways to refresh the Application Server: using the **refresh.exe** command line version or the graphical user interface (GUI), **RefreshApp.exe**.

Upon startup, the Application Server reads metadata and Registry information and caches it for use during normal query processing. The metadata that is read includes:

- The Windows Registry
- The ***config_master*** table in the EpiMeta database
- Aggregate navigation information.

Aggregate navigation is the process by which the Epiphany query machinery determines the optimal aggregate table to use to satisfy an application's request for information. (The *query machinery* is the component of the Epiphany Application Server that communicates with EpiMart and actually issues SQL statements against the DBMS.)

- Time navigation information.

Time navigation is similar to aggregate navigation information. When issuing the backlog queries, the query engine uses time navigator to find the most optimal time path to calculate the backlog.

- The templates used to render Clarity, Relevance, and Momentum result (report/chart/list) pages
- Security Information
- Ticksheet information

When any of the following events occur, you need to restart or refresh the Application Server:

- After an extraction, either the *current_datamart* has changed or there has been a change in a table that is used in a dynamic listbox filter.
- If you Restart the SQL Service *MSSQL Server* on the server on which the Epiphany database (EpiMart) resides while your Application Server was running against that database, then you must restart the Application Server.
- EpiCenter Manager was used to alter the EpiMart.
- Aggregates have been built or rebuilt since the last time the Application Server was started.
- EpiCenter Manager or Web Builder was used to add or modify a ticksheet, measure, or dataset.
- The Windows Registry entries under the following entry have changed since the last time the Application Server was started:

HKEY_LOCAL_MACHINE/Software/Epiphany/instance_name

Refreshing the Application Server

- EpiCenter Manager or Security Manager was used to change security permissions for a user or group.

Note: The fact that a template file has changed does *not* require a restart or refresh of the server.

If you have determined that you need to refresh the Application Server, there are several ways to proceed. The simplest way to refresh the server is to stop and then restart the Application Server via the Services Control Panel. However, this requires manual intervention, and so would not be suitable for programmatic refresh (after an extraction, for example). This method also interrupts anyone currently using the system.

Another way to refresh the server is by using the **refresh.exe** program that was installed in your **installdir\win32** directory. This program sends a message to the Application Server instructing it to re-read all of the necessary information mentioned above. The Application Server re-parses all of the template files (in case the TemplateDir Registry key was changed) and re-initializes the Security objects.

Refreshing will not interrupt the requests that are currently running on the Application Server. After the refresh has been completed, however, users who were previously logged on will need to log in again. This is because the security restrictions that affect all users' rights have been changed.

You may invoke the **refresh** program via a DOS command-line or open the RefreshApp dialog box and enter command options. Instructions for both methods are given below.

The Refresh Command Line

The **refresh** command-line syntax is:

refresh localhost port username password

where:

localhost Specifies the name of the machine on which the Application Server is running.

port Specifies the port number on which the Application Server is listening.

Username and password
Specify a valid administrative account.

These are the same values required for an end user to log into the top-level Epiphany Web page. In general, any account that is defined in the Security folder of the EpiCenter Manager application will work. See “Security” on page 200 for more information.

After the **refresh** application has established a connection to the Application Server and sent the appropriate messages, it displays the following message and waits for a response:

Sent the REFRESH instruction to localhost

Normally, the Application Server takes about 30 seconds to refresh before returning an acknowledgment. (Times may vary based on the size of your EpiMeta database and the speed of your network, and other considerations.) Upon receiving this acknowledgment, the **refresh** program displays:

```
Refresh [Build 3.4.0.1035]
-----
Connecting to localhost:8081
Sent the REFRESH instruction to localhost
The refresh SUCCEEDED.
REFRESH operation took 18827 ms.
```

Invoking RefreshApp

Otherwise, **refresh** indicates that it has failed by outputting the response that it does receive, or by indicating that it has failed. For example, here is the output of a negative interaction in which the user/password failed.

```
Refresh [Build 3.4.0.1035]
-----
Connecting to localhost:8081
Sent the REFRESH instruction to localhost
The refresh FAILED because the username/password combination was
invalid.
REFRESH operation took 3365 ms.
```

(The **refresh** program always displays the amount of time that it has taken.)

Invoking RefreshApp

RefreshApp is the GUI version of the **refresh** executable. You may keep the RefreshApp window open and on your desktop during debugging.

Follow these steps to use RefreshApp:

- Step 1:** Choose the Refresh application from the Start menu:
Programs\Epiphany\RefreshApp.
- Step 2:** For the AppServer name, enter the name of the machine on which the Application Server is running.
- Step 3:** For the AppServer Port, enter the port number on which the Application Server is listening.
- Step 4:** For the username and password, specify a valid NT account that has access to the Epiphany Application Suite.

These are the same values required for an end user to log into the top-level Epiphany Web page. In general, any account that is defined in the Security folder of the EpiCenter Manager application will work. See "Security" on page 200 for more information.

Step 5: Click Refresh to start the application.

Messages display in the bottom of the dialog box that tell you whether the Refresh succeeded or failed (and the time that this event occurred).

Step 6: In case of failure, click View Log to determine what caused the failure. The information that is normally printed to the screen during the execution of **refresh.exe** is displayed in this window.

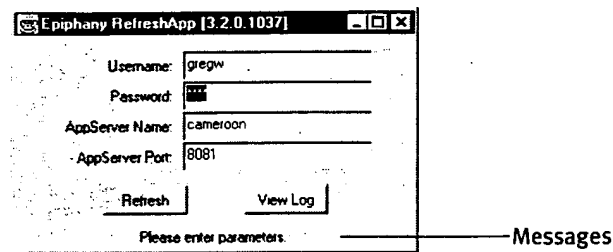


Figure 77: EpiApp RefreshApp Dialog Box

THE EPIAPPSERVICE PROGRAM

The EpiAppService program (**epiappservice**) is the program that Windows NT uses to run the Application Server as an NT Service. You will use EpiAppService to—

- delete an Epiphany Application Server service.
- change the parameters of an Epiphany Application Server service.
- add a new Epiphany Application Server service; for example, create a service because the installation failed, or to install another instance.

Command Syntax

Command Syntax

The `epiappservice` command syntax is—

```
epiappservice -s servicename action
```

where *action* is one of the following parameters:

-C Creates a service. Registers a new service with the NT system. You must use the **-E** option with this option to specify the executable to use as the NT Service; for example:

```
EpiAppService -S "instance" -C -E "program name"
```

When you are configuring an Epiphany Application Server, the command line will look like:

```
EpiAppService -S "instance" -C -E "C:\Program  
Files\Epiphany\instance\win32\EpiAppService.exe -S instance"
```

If the instance name has spaces in it, then you should use single quotes inside of the outside quotes. Do not use spaces within the instance name.

Run the EpiAppService program to create the service, which is also named EpiAppService. It is invoked with the **-s** option to specify the instance name. (The EpiAppService program is both the program that should be run as a service, as well as the program used to register the service.)

-E *cmd* The command to associate with the service.

- DEP *service*** Allows you to specify a service on which the Application Server depends. The Service Manager will always load this dependency before loading the Epiphany Application Server. The service name supplied as an argument to this option must exactly match the service name used by the service in question, usually the Microsoft SQL Server.
- D** Deletes the service specified via the **-s** option from the NT Service Registry. When this service is deleted, only the entry in the NT Registry for the service is deleted; no executables are deleted. Use this to remove unused Application Server instances.
- G** The launch handler for the service. (Go.)
- T** Starts a service. The installation program uses this option to start the Web server (which must be shut down during installation). It starts the service specified via the **-s** option, which is equivalent to clicking the Start button in the Control Panel\Services dialog box to start the desired service. You may use this option when you cannot access the Control Panels (for example, during RCMD or pcAnywhere remote access).
- P** Stops the service specified via the **-s** option. (See the **-T** option.) The installation program uses this option to stop the Web server during the installation.
- K** Checks if the service specified via the **-s** option is running. The program returns a zero return code if the service is running, and a non-zero return code if it is not.
- ?** Displays online help, a description of the command options.

EppiAppService Command Examples

EppiAppService Command Examples

Examples of the most common invocation of **eppiappservice** follow:

- Creation of a service because the installation failed, or you wish to install another instance. If the database server is on a different machine from the Application Server:

```
EpiAppService -S instname -C -E "C:\Program Files\  
Epiphany\instname\Win32\EpiAppService -S instname"
```

If the database server is on the same machine as the Application Server:

```
EpiAppService -S instname -C -E "C:\Program Files\  
Epiphany\instname\Win32\EpiAppService -S instname" -DEP  
MSSQLServer
```

- Deletion of an old service.

```
EpiAppService -S instname -D
```

The Application Server's Registry Keys

All Registry keys used by the Epiphany Application Server are located in **HKEY_LOCAL_MACHINE/Software/Epiphany/Instances/instance_name** where **instance_name** specifies the name of the instance you entered during the Epiphany software installation. This directory contains the following keys:

AppServerHost The name of the machine on which the Application Server is running. The default is the *localhost*.

ApplicationServerPort The port number on which the Application Server is listening for connections. The default is 8081.

AppServerLogVerbosity

This parameter specifies the degree of logging (verbosity level) that the Application Server performs when logging to the database. If the value is 0, no logging is performed. If the value is 1; logging is performed.

AppServerQueryTimeout

The number of seconds before the Application Server automatically terminates long-running queries. A query that requires more than this number of seconds to process will be terminated in order to prevent the exhaustion of system resources and run-away queries from bringing down the server.

AppSessionTimeout

The value in seconds for the lifetime of an idle session. If a session has been idle for this many seconds, then it is removed from the cache.

Build

The version number of the Application Server. This value should be 3.4.

ChartsOutputDir

The directory (full path) in which the *.epc chart files will be stored.

DatabaseName

The name of the EpiMeta database.

DatabaseServer

The name of the database server on which the EpiMeta database is located.

DatabaseUsername

A username for logging into both the EpiMeta and EpiMart databases.

DatabasePassword

The password that corresponds to the DatabaseUsername account.

DatabaseType

Specifies the type of the database.

The Application Server's Registry Keys

Description	A textual description of the instance.
DSN	The name of the DSN that is used to connect to the EpiMeta database.
InstanceRootDir	The root directory into which the Epiphany Application Server has been installed.
ProxyLogFile	(Optional) Enables proxy logging. For example, setting HKEY_LOCAL_MACHINE\SOFTWARE\Epiphany\Instances\capri\ProxyLogFile to C:\proxy.log will create a proxy.log file if it does not already exist, and log information for every proxy submit to the Application Server. If an invalid path is specified in this ProxyLogFile Registry key, no logging will be performed.
SystemLogDir	The directory in which the Epiphany Application Server log files are written.
SystemLogDirWebpath	The name appended to http://machinename/instance_name/ that informs the Web server of the log files' locations.
TempDirGarbageLifetime	The lifetime in seconds of a temporary or log file created by the Epiphany Application Server. After this specified number of seconds from the creation date, a temporary file, log file, or *.epc chart file will be erased by the Temporary FileManager in the Application Server. Garbage collection occurs at Application Server startup and periodically when there are free cycles.
TemplateDir	The directory in which the Epiphany Application Server templates are stored.

Driver	The JDBC driver used to connect to an EpiMart database. The default is connect.microsoft.MicrosoftDriver . <i>Do not change this value.</i>
SecurityClass	(Optional) The Java class to use for security authentication. If it is not specified, the com.epiphany.security.EpiNTLogon class is used.

THE EPIPHANY PROXY

The Epiphany proxy (**Epiphany.dll**) is an ISAPI application that mediates the requests and responses between the IIS Web server and the Epiphany Application Server. All user requests for Epiphany pages are directed to the proxy **http://machinename/scripts/instance_name/Epiphany.dll**.

This proxy bundles the request into a package that conforms to a strict Epiphany format and sends the package to the Epiphany Application server through a TCP/IP socket. The proxy uses the Windows Registry to find the Epiphany Application Server. In particular, the proxy parses the *instance_name* out of the requesting URL. It then opens the AppServerHost and AppServerPort with that instance's Registry tree, and uses this information to connect to the Application Server. The Application Server processes the request and sends a result back to the proxy. The proxy displays the result in the user's browser.

Note: The *instance_name* in the URL must match the *instance_name* as defined in the Windows Registry (configured from the installation program). This allows one proxy to direct requests to several different Application Server instances. This is of value when you want to have two versions, such as a release and a test instance.

Proxy Logging

The **Epiphany.dll** ISAPI proxy supports rudimentary logging. If you suspect that the proxy is not passing all parameters to the Application Server, you can enable logging of all parameters that the proxy submits.

Proxy Logging

Proxy logging, which is optional, is a diagnostic tool for identifying problems, not a run-time logging facility. Because the log file may grow arbitrarily large if proxy logging is always enabled, use it only in the event of a suspected problem with the **Epiphany.dll** proxy.

An example of a proxy log file:

```
Processing new request. Header: mGET q p v0.8 oHTTP/1.0 nzhenya
P80 r192.0.0.147 aGodzilla/4.04 [en] (WinNT; I ;Nav) u
U/scripts/capri/Epiphany.dll S/scripts/capri/Epiphany.dll
Requested dispatched. Request data length was 0
Processing new request.
...
Request data length was 72
```

The log file consists of blocks, with each block representing a submit to the proxy DLL. Each new submit starts with `Processing new request. Header:` and is followed by the header. The header consists of header items, each beginning with a letter that describes the type of item, followed by the value of item.

The first letter of the header item is defined below.

m	method POST or GET
q	query string (used with GET)
v	version
o	HTTP protocol
n	server name
P	server port
r	remote IP address
a	user agent

u	user name (if Web server is using authentication)
U	URL
S	script name

The header will be followed by the data section if the submit was of the type POST. First, a number of bytes is printed (172), then the actual data, which should be the same size as the number of bytes printed. Finally, if the submit was successful, the following line is printed:

Requested dispatched. Request data

If the submit failed, then the log file should read:

FAILED TO DISPATCH REQUEST DUE TO A SOCKET ERROR.

APPLICATION SERVER LOGGING

Most components of the Application Server maintain a log file of their activities. This section describes each of these logs, their directory location, and their diagnostic use.

The components that generate logs and the contents of those logs are briefly described below (and in more detail later in this section).

Server	The com.epiphany.server.Server class generates a log file of connections made to the Application Server. This file contains the time at which a connection was accepted, the number of that connection, the time at which the request was finished, the total time required by the request, and certain messages printed to the log during the processing of that request. The latter category includes the session ID of the connection, the template used to process that request, the number of bytes generated in the response, exceptions that might be
---------------	---

Log File Location

generated during the processing of the request, or the user name used to log into the server.

SecurityManager

The **com.epiphany.security.SecurityManager** class generates a log that contains the users and groups in the system. It also captures login and sync-up information.

StorageManager

Logs all function calls, SQL, exceptions, and errors for the storage subsystem.

Clarity, Relevance, or Momentum Query Log

Each Clarity, Relevance, and Momentum request generates a log that contains the submitted timesheet's parameters, information about how the timesheet was parsed, all of the SQL statements executed during the processing of the request, and information about how long the processing took.

Log File Location

In the Windows NT Registry on the Application Server machine, the Registry key that specifies the location of all log files generated by the Application Server is named:

HKEY_LOCAL_MACHINE\Software\Epiphany\Instances\instance_name\SystemLogDir

Log File Naming Conventions

All log file names begin with a prefix that indicates the date and time when the log file was first created. The format is as follows:

YYYY-MM-DD_HH-MM-SS-MSname.txt

where YYYY stands for the year, MM for the month, DD for the day, HH for the hour, MM for the minute, SS for the second and MS for the millisecond. *name* can be one of the following: SRV, SECURITY, STORAGE, or QM_sessionid. The logs of queries for Epiphany applications have the suffix QM_sessionid.

The Server Log

When you start the Application Server, the server object creates a log immediately after the Registry has been opened and read. The Registry must be opened first because it contains the SystemLogDir Registry key that specifies exactly where the log files are stored.

```
**** Epiphany AppServer Acme awaiting connections... ****
Mon Jan 08 14:48:50 PDT 1999:
+++++ Accepted [1] @ [897342530272]
----- Dispatched [1] @ [897342530322]
Free/Total Memory: 165064/3297272
Setting to default template (no app): toplevel
No session exists/created. Displayed login screen.
4865 bytes generated.
----- Finished [1] @ [897342530773]
501ms

Mon Jan 08 14:48:59 PDT 1999: +++++ Accepted [2] @ [897342539215]
----- Dispatched [2] @ [897342539245]
Free/Total Memory: 719832/3342328
Template passed in: toplevel
Got EP_USER: lslater
Adding session: EPIPHANY\lslater|192.0.0.12
5184 bytes generated.
----- Finished [2] @ [897342539615]
400ms
```

The Security Manager

Note that the server writes this information for each socket connection it accepts:

Date: +++++ Accepted [connection number] @ [milliseconds since epoch]

For example:

Sun Feb 03 17:54:16 PDT 1999: +++++ Accepting [1] @ [894243256100]

The *Date* is written using the **java.lang.Date.toString()** method. Hence, it uses the default formatting on the server machine. The connection number is the numerical ID of the connection that was accepted. Numerical IDs are assigned in increasing order, starting from 1. The *time in ms since epoch* is determined using the **System.currentTimeMillis()** method. It represents the number of milliseconds that have elapsed since January 1, 1970.

During the processing of the request, certain messages might be printed to the server log if no other log is available. This is the case for the TemplateServer and the EpiAdminServer.

After the request has been started, the server indicates the end time of the request in this format:

Finished [<connection number>] @ [<milliseconds since epoch>] 400ms

The Security Manager

The Security Manager log first prints out all of the groups recognized by the system. Each entry in this first section contains the group name, the key in the database, and a list of ticksheets accessible by members of the group. For example:

```
Groups: {Administrators=
Group: Administrators / key: 4 / world save query access: 1
Access List:
Assigned ticksheets:
ExecutiveBestWorst,ExecutiveProfiles,ExecutiveProjections,
ExecutiveTrends,Expense,
```

The next section of this log file contains a list of all the users recognized by the system. Each entry contains a user name, the groups to which the user belongs, and the list of ticksheets that the user has permission to view. Note that the ticksheet list is mostly empty since all users belong to at least one group, and therefore inherit ticksheet permissions.

Save and Restore Manager

The Save and Restore Manager logs all function calls, SQL, exceptions, and errors for the storage subsystem. If you cannot save a file, this is a good place to check.

Epiphany Applications

Note: If the **log** link at the bottom of an HTML Clarity/Relevance/Momentum result page does not take you to a log, make sure that the SystemLogWebDir Registry key refers to an alias that has been configured on the Web server. Also make sure that the directory and files have correct read permissions.

Each Epiphany application creates its own log file before performing any non-trivial processing. The log contains the version of the Application Server and the date/time of the request. The second line begins with `APPLICATION parameters:` and subsequent lines contain all of the name-value pairs submitted to the Application Server for this request. Multiple values submitted with the same name are shown separated by commas. A blank line follows, and then the Application filters, dimensions, and measures that could be parsed from the submitted ticksheet's parameters are logged.

Next, the system logs all of the SQL statements needed to compute the answer to the request, the amount of time in milliseconds to process the query, and the number of rows returned for all SQL queries.

The log ends with the `CLEANUP` message.

APPLICATION SERVER SECURITY

Authentication, which is performed outside of the Epiphany system, does not capture password information. This external authentication requires an authentication module. For example, the NT authentication module authenticates users with an NT domain controller. The Security Manager inside the Application Server loads the authentication module specified through the SecurityClass Registry key at initialization. Each authentication module supports a fixed API that includes methods to authenticate users, add new users to the Epiphany system, and sync-up outside information on the user (such as group memberships) with Epiphany metadata.

Currently, Epiphany provides two authentication modules: the NT authentication module EpiNTLogon, and an insecure authentication module called EpiPassThruLogon. Use EpiPassThruLogon only for testing and debugging. Optionally, you can add an LDAP authentication module, X.500 module, and other similar modules.

The optional Registry key SecurityClass under the *instance_name* Registry directory controls which security module is loaded. You must specify the full class path to the security module. If this key is omitted, the system uses the default security module **com.epiphany.security.EpiNTLogon**, which uses NT to perform user authentication.

The means by which the authentication information (username and password) reaches the Application Server is as follows. Users log into the Epiphany system either through a login template or through a Web server authentication mechanism, such as Basic Authentication or NTLM.

When the Application Server receives the user name and password, it calls the Security Manager to log in the user. The Security Manager loads an authentication module, and attempts the login process. If the process is successful, depending on the authentication module, one of the following steps is taken:

- If the user exists in the EpiMeta database, user group memberships outside of the Epiphany system will be synched up with group memberships in the EpiMeta database.

- If the user does *not* exist in the EpiMeta database, but is authorized to use the Epiphany system, then a user account will be created in the EpiMeta database. User group memberships outside of the Epiphany system, such as NT, will be synced-up with group memberships in the EpiMeta database. For example, assume your EpiMeta had a group named EPIPHANYUSERS configured with access to all of the ticksheets. If a user name Joe (a valid NT user who belongs to the NT group EPIPHANYUSERS) supplies a correct password, then Joe will be automatically added to the EpiMeta metadata as a user who belongs to the Epiphany group EPIPHANYUSERS.

Only group memberships for a group whose Group Definitions dialog box in EpiCenter Manager has the Synchronize option selected will be synced-up. Sync-up occurs if:

- the user who logs in is 1) a member of a Group X outside of the Epiphany system; 2) Group X exists inside the Epiphany system; and 3) the user is not a member of Group X inside the Epiphany system. Sync-up will create a group membership to Group X for this user inside the Epiphany system.
- the user who logs in is 1) a member of a Group X inside the Epiphany system, but 2) the user is not a member of this group outside of the Epiphany system. Sync-up will remove a group membership from the Epiphany system.

If the sync-up process requires a creation of a new group membership for a user, certain access rights are set on this membership. The ability to save queries has the access rights of Save Group/Default, and global-level save access is Inherit. (See “Security” on page 200 for more information.)

For a group to be the same in EpiMeta and the NT domain, it needs to have the identical name (case sensitive) in both. The group name in the NT domain includes the domain name, such as **Epiphany\Sarah**. You need to make sure that the group name in the EpiMeta includes the domain name for that group.

When the user is authenticated and user information is synced-up, Security Manager determines if the user is authorized to use the Epiphany system. The user is so authorized if he or she belongs to at least one group in the Epiphany system. (The user must also must log in with a valid password.)

Authentication Modules

Important: An authenticated user is authorized to use the Epiphany system if and only if that user is a member of at least one group in the Epiphany system after sync-up.

Authentication Modules

The following authentication modules are included with the Epiphany software.

Module	Class Name
NT (default)	com.epiphany.security.EpiNTLogon
Pass through	com.epiphany.security.EpiPassThruLogon

- NT (default)
NT domain authentication. NT groups are imported into the Epiphany system through EpiCenter Manager. As mentioned, these groups need have the Synchronize option selected in the Group Definition dialog box, which means that memberships to those groups will be synced-up. Users who belong to those groups can log into the Epiphany system. When a user logs in, his or her NT group memberships are synced-up to the EpiMeta database. It is also possible to create Epiphany-only groups inside EpiCenter Manager by simply not selecting the Synchronize option. Thus, the Epiphany administrator may create arbitrarily complex permission hierarchies using EpiCenter Manager, independent of the manner in which NT domain security is set up.

- Pass through (*for in-house debugging purposes only; should never be used at a customer site after the initial Epiphany system setup*)

This is an insecure authentication that has no password. This is an Epiphany-only development module that ignores NT authentication all together. You do not need passwords in this model: specify your user name exactly as it appears in EpiCenter Manager (it must include a domain name if such exists), and you will be logged in. There is no sync-up process. That means that an authenticated user that does not exist in the EpiMeta database will not be allowed to use the Epiphany system. Epiphany groups and users are created and managed through EpiCenter Manager.

Authentication Module Tips

- If you want to use NT-related authentication modules, make sure that **EpiNTLoginJNI.dll** is in your system path.
- Users that do not belong to any groups in the Epiphany system are not allowed to log in. An error message that says that user is authenticated but not authorized to use Epiphany system is displayed whenever an unauthorized but NT-authenticated user logs into the Application Server. User memberships may be adjusted upon login if the user is a member of synchronized groups in Epiphany system. A user must be a member of at least one Epiphany group after the synchronization process completes.
- The optional Registry key SecurityClass (located in the *instance_name* Registry directory) controls which security authentication module is loaded. The full class path to the security module must be specified as the value for this key. If this key is omitted, the default security module is **com.epiphany.security.EpiNTLogon**, which uses NT to perform user authentication.
- The **Toplevel.template** is the template that appears immediately after a user logs into the Epiphany system. The name of this template is configurable with the an optional ToplevelTemplate Registry entry in the **Instances** directory. You can load the **company.template** instead of **toplevel.template** by substituting your company name in ToplevelTemplate=*company*.

Authentication Modules

- Depending on how IIS security is set up, one of the following situations occurs upon login:
 - If Allow Anonymous authentication is enabled, the login dialog box is displayed when the user attempts to use the system for the first time, or the user session times out. It is almost always sufficient to specify the username only. The domain name is located automatically.
 - The search order that authentication uses to find the user account is as follows. First, the authentication mechanism looks in the local machine's Security Access Manager (SAM), then it checks with the primary domain controller, and afterwards checks with trusted domains. If there are multiple users with the same name between domains and/or a local machine that runs Application Server, specify the full user *name-domainname\username*, or the *local_machine_name\username* if the user logs in from a local machine's Security Access Manager (SAM).
 - If Basic Authentication is enabled, the browser displays a login dialog box when the user attempts to access the Epiphany system for the first time. However, when the user's session times out, no re-login is required. The user is automatically logged in again, and a new user session created.
 - If NTLM authentication is enabled, Internet Explorer automatically performs authentication of the user without displaying a login dialog box, although Netscape displays it. If Basic Authentication or NTLM is on, the login dialog box does not appear in the Web browser.

Important: Do not create your own domains. Doing so introduces multiple domains, with the Epiphany machine in one domain and the user accounts of the Epiphany system in another domain. In order for the authentication module **com.epiphany.security.EpiNTLogon** to work properly, a two-way domain trust between the Epiphany domain and the customer domain is required.

If you set up a new domain for the machine that runs the Epiphany Application Server, set up a two-way trust and name the machine and the domain differently. In general, do not use the same string for domain names, machine names, and user names.

- The following applies to Synchronized groups:
EpiNTLogon requires the NT domain for lists of global and local groups. The names of these groups will be matched to the names of the groups in the EpiMeta. A group name will be matched if its domain name and group name match. Therefore, group *foo* will not match to group *EPIPHANY\foo*. (The match is case insensitive for both group name and domain name.) If a user is a member of an NT group *EPIPHANY\foo* and EpiMeta has a group called *epiphany\FOO*, there will be a match.
- If a user is a member of a local group and the group has a global group as a member, the global group will not be picked for synchronization process. Only the groups for which the user is an immediate member are considered for synchronization process.
- If a user logs in with an account that is local to the Application Server machine, then a membership to a special group called None is automatically retrieved from the machine's SAM. (Every user in the local SAM has a membership in a special global group called None although this group does not exist on the machine.) For this reason, do not create synchronizable groups called None in EpiCenter Manager.
- Avoid having the same name for domain names, machine names, and user names. For example, if the Application Server runs on the machine *foo*, and the user called *foo* attempts to log in, access may be denied. If the Application Server is running on a machine named *foo*, and a user named *foo* logs in from the primary domain, or from the local Security Access Manager (SAM) of the machine *foo*, authentication will succeed. If user *foo* logs in from a trust domain, however, the authentication will fail. The only way to log in as *foo* from another domain is to give the full name for the user account upon login: *domainname\foo*.

Administrator Groups

ADMINISTRATOR GROUPS

Epicenter Manager users can make any group an administrator group, and there can be multiple administrator groups in the system. If the user belongs to such a group, that user has special powers when it comes to report/folder and ticksheet access.

Administrator users can save, overwrite, create new, change properties and permissions on any non-special and non-hidden file or folder.

Note: Special folders are top-level folders, such as Public and All Users, or user/group folders or default folders. Hidden folders are the MailTo folder; hidden files are clipboard files.

Administrator users have access to all ticksheets in the system. In the future administrator users will be able to perform more powerful operations than regular users.

APPENDIX A

EPIPHANY MACROS

This appendix describes the Epiphany-supplied system calls and SQL macros.

Note to sites that use their own database types and SQL macros:

The various fact semantic types (such as Transactional/Statelike) use SUM() operators over the fact columns. New tables are created via SELECT INTO from the result. With decimal data types such as NUMBER(16,4) the precision of the resulting column tends to grow under SQL Server until the column takes up many bytes in the fact table. For this reason, both FACTQTY and FACTMONEY map to the MONEY database type in SQL Server.

SYSTEM CALL MACROS

For the most part, you will use Epiphany system call macros for the transfer of information related to the locations of files and database logins from the EpiMeta database to system calls. Many systems calls such as **mv** and **mkdir** are unable to read a database, and few system calls will be able to read Epiphany-created structures. Rather than leaving each system call to its own means to determine its appropriate information, EpiChannel may pass this information to the system call on the command line.

System Call Macro Syntax

System Call Macro Syntax

The syntax for system call macros can be either

`$$VERB[arg1, arg2, ...]`

or

`$$VERB`

Unless stated otherwise, the arguments are the names of roles defined with the job. You may use multiple arguments in most cases, which results in an expansion of each argument. Note that the expansion of both of the following is the same:

`$$verb[arg1, arg2]`

`$$verb[arg1] $$verb[arg2]`

If the verb does not match one of the special words, no expansion of that verb occurs; for example:

`echo $$notAverb`

expands to

`echo $$notAverb.`

If the argument is a role name, and the job does not define that role, the system call is considered to have an error. The following table describes the Epiphany system call macros.

Note: The Usage column in the following table stands for expected frequency of use.

Table 3: Epiphany System Call Macros

Macro	Purpose	Usage	Description
AGG	Child Procs	High	The name of the Aggbuilder executable in \$SEPIBIN. For example: \$SAGG \$SEXC_ARGS -j \$\$JOB_NAME

Table 3: Epiphany System Call Macros

AGGVERIFY	Child Procs	Low	The name of the aggverify executable in \$\$EPIBIN.
APPSERVERHOST	Registry	Low	The value of this Registry variable.
APPSERVERPORT	Registry	Low	The value of this Registry variable.
CHARTSLOGFILE	Registry	Low	The value of this Registry variable.
CHARTSOUTPUTDIR	Registry	Low	The value of this Registry variable.
DATABASE	Database Login	High	<p>Translates to the name of the database or instance. For example:</p> <pre>isql /S \$\$SERVER[Tests] /U \$\$USER[Tests] /P \$\$PASSWORD[Tests] /d \$\$DATABASE[Tests] /w 300 /i /Q "gen_tests_run"</pre>
DBVENDOR	Database Login	Medium	Translates to the vendor of the database technology.
DEBUG_LEVEL	Command Line	Low	Translates to the current verbosity level of EpiChannel. Use this to pass EpiChannel's verbosity onto the subprocesses it spawns via system calls.
DIRNAME	File ID	Medium	<p>Translates to the directory name of the data store, without the last filename component and without a trailing slash. If the role is working dir, the directory name's last component is a unique subdirectory generated for this particular run of EpiChannel. For example:</p> <pre>echo DIRNAME is \$\$DIRNAME[Working Directory]</pre> <p>(but with no arguments it is \$\$DIRNAME).</p>

System Call Macro Syntax

Table 3: Epiphany System Call Macros

DSN	Database Login	Medium	Translates to the ODBC connection string for the database. This string may be generated even for databases accessed using native API's.
EPIBIN	Child Procs	Medium	The name of the win32 directory under the InstanceRootDir Registry variable.
EXC	Child Procs	High	The name of the extract executable in \$\$EPIBIN.
EXC_ARGS	Child Procs	High	The recognized portions of the extract command line.
EXC_CMD	Child Procs	Medium	The extract program and its arguments other than job name. You can use this to fire sub-extract runs. For example: \$\$EXC_CMD -j performance
EXCVERIFY	Child Procs	Low	The name of the excverify executable in \$\$EPIBIN.
FILENAME	File ID	Medium	Translates to the filename of the data store. If the role is Working Dir, the file name is the name of the EpiChannel log file. For example: echo FILENAME is \$\$FILENAME[Working Directory] (but with no arguments it is \$\$FILENAME).
INSTANCE_NAME	Registry	Medium	The name of the instance's Registry subtree.
INSTANCEROOTDIR	Child Procs	Low	Value of the InstanceRootDir Registry variable.
ISS	Database Login	Low	Translates to a number associated with this source of information.

Table 3: Epiphany System Call Macros

JOB_NAME	Child Procs	High	The name of the current job.
PASSWORD	Database Login	High	Translates to the password associated with the database login.
PATH	File ID	Medium	Translates to the directory name and file name placed together (in the DOS file path format).
PROGRAM_NAME	Child Procs	Low	The name of the current extract program.
REGISTRY_EIPATH	Registry	Low	Name of the Epiphany Registry key.
REGISTRY_ROOT	Command Line	Low	Translates to the Registry key used by EpiChannel.
SERVER	Database Login	High	Translates to the database host server name (the machine's name) in the case of Microsoft SQL Server, or the SQLNet ID (sid) in the case of Oracle. SERVER and SQLNET are identical in behavior and can be interchanged.
SQLNET	Database Login	High	Translates to the database host server name (the machine's name) in the case of Microsoft SQL Server, or the SQLNet ID (sid) in the case of Oracle. SERVER and SQLNET are identical in behavior and can be interchanged.
USER	Database Login	High	Translates to the username associated with the database login.
VERSION	Database Login	Low	Translates to a release number or string associated with this database's installation.

EPIPHANY SQL MACROS

Major database vendors have developed proprietary extensions to SQL that customers may choose to use in their SQL code because of their features. Epiphany also provides SQL macros for SQL Epiphany products. These macros, which are available as an option to customers, attempt to be database vendor-independent.

Epiphany supplies SQL macros to resolve these issues:

- Support coding of SQL statements that works with the syntax of multiple database engines.
- Support SQL that adapts to the structure of the star schema as defined via the EpiCenter Manager.
- Support extraction of contiguous but non-overlapping subsets from those tables that have dates or ascending identification fields.
- Control the physical characteristics of an Oracle EpiCenter.

Oracle-specific SQL Macros

Oracle-specific SQL macros control the physical characteristics of an Oracle EpiCenter. Oracle tables are stored in a logical entity called a tablespace. Because of the various size requirements of EpiCenter objects, such as fact tables and indexes, dimension tables and indexes, EpiCenter allows you to configure which tablespace is used for each object type. The Oracle-specific SQL macros are translated into tablespace names.

Table 4: Oracle-specific SQL Macros

Macro	Definition
\$\$DIMTABLESPACE	Used for dimension tables (including aggregates and mini-dimensions).
\$\$DIMINDEX_TABLESPACE	Used for dimension indexes (including those on aggregates and mini-dimensions).

Table 4: Oracle-specific SQL Macros

Macro	Definition
\$\$FACTTABLESPACE	Used for fact tables (including aggregates and clusters, as well as temporary objects used during semantics).
\$\$FACTINDEX_TABLESPACE	Used for fact indexes (including those on aggregates and clusters).
\$\$TEMP_TABLESPACE	Used for all Application Server temporary tables (those needed for query post-processing at runtime).
\$\$METATABLESPACE	Expands to the name of the tablespace to use for metadata tables on your system.

Normally, the values for these macros are set to match the names of the tablespaces as they are created by the Oracle initialization script provided by Epiphany as described in the *Epiphany Installation Guide*.

To use alternate names for these tablespaces, you need to run SQL statements such as the following against your EpiMeta database.

```
update translation_Actual set actual_string =
'your tablespace name' where store_type = 'Oracle' and
translation_string = 'FACTTABLESPACE'
```

Vendor-independent Macros

Although it is possible to avoid proprietary extensions to SQL, the features they offer may be necessary. As a result, consumers of SQL inherit problems raised by these differences unless they decide to bind themselves to a single database vendor. For example, a function called NVL in one database is equivalent to a function called ISNULL in another database.

SQL Macro Usage

Epiphany supports multiple database vendors, and Epiphany-executed SQL statements are a major consumer of SQL features. The Epiphany vendor-independent macros serve to provide some degree of isolation from database vendor differences. Epiphany uses these SQL macros in its own SQL so that it does not need to support different “codelines” of SQL.

Although these macros are available to use by Epiphany customers, their use is optional. Customers concerned only with a single database vendor might avoid all use of these macros. However, customers who use multiple vendors now or plan to in the future may choose to use the Epiphany SQL vendor-independence macros in the SQL statements they create.

SQL Macro Usage

An example of both preferred and less preferable SQL macro usage follows:

Preferred:

```
where COLUMN_FILTER[outfitting_order~,~outfitting_order_key~,~oo]
```

Less preferable:

```
where COLUMN_FILTER[ outfitting_order ~,~ outfitting_order_key  
~,~ oo]
```

The following tables provide a brief explanation of each SQL macro:

- Table 5, “Adaptive SQL Macros,” on page 265
- Table 6, “Extraction Set Identification Macros,” on page 266
- Table 7, “Smart Extraction Macros,” on page 269
- Table 8, “Vendor-independent Macros,” on page 270
- Table 9, “Testing Macros,” on page 284

SQL Macro Notes

The following applies to SQL macro syntax:

- You can determine the “real” translations for most Epiphany macros by issuing the following SQL in an EpiMeta database:

Select * from translation_actual

- For usage examples of most of the SQL macros, see the initialization file **templates.sql**.
- The syntax for Epiphany SQL macros may be either **\$\$Macro[arg1~,~ arg2~,~ ...]** or **\$\$Macro**
- The ~,~ is used to separate the arguments if there are multiple arguments.
- If an argument consists of multiple elements, separate the elements using tilde-comma-tilde. For an example, see the macro **CREATE_INDEX_IF_NOT EXISTS** given in Table 8, “Vendor-independent Macros,” on page 270.
- In some of the macro examples, column spaces have been added near the brackets and near the ~,~ entries for better formatting. In reality, any spaces that are present would be forwarded into the final SQL, so the best coding practice is to avoid them.
- The Usage column in the following tables stands for expected frequency of use.

Table 5: Adaptive SQL Macros

Macro	Usage	Description
CURR	High	Expands the _A or _B suffix of the currently active tables. Allows you to reference the active or new EpiMart tables.
NEXT	High	Expands the _A or _B suffix of the not-currently-active tables.

Table 6: Extraction Set Identification Macros

Macro	Usage	Description
COLUMN_FILTER [table_name ~,~ column_name ~,~ alias_name]	High	Expands to a "one-sided" SQL comparison expression that requires that alias_name.column_name be greater than or equal to the value in table table_name column column_name as of the start of the last run. This extracts "everything since last time." Although the alias name is optional; you should use it.
COLUMN_LAST_VALUE [tbl ~,~ col]	High	Expands to the value of this column as of the start of the last run. This expands to a value, not to an expression, allowing you to make your own expressions.
COLUMN_RANGE_FILTER [table_name ~,~ column_name ~,~ alias_name]	High	Expands to a "two-sided" SQL comparison expression that requires alias_name.column_name to be greater than or equal to the value in table table_name column column_name as of the start of the last run, and less than or equal to this value as of the start of the current run. This extracts "everything since last time, but not including that data that changes while the extractions are running."

Table 6: Extraction Set Identification Macros

DATE_FILTER[column_name]	High	<p>Expands to a “one-sided” SQL comparison expression that requires the column to be greater than or equal to the “current date/time” as of the start of the last run. This extracts “everything since last time.”</p> <p>No table or alias arguments are available to the macro. If the column needs to be qualified, just add the qualifications in the first argument. For example:</p> <p>and \$\$DATE_FILTER[oo.date_key]</p>
DATE_RANGE_FILTER [column_name]	High	<p>Expands to a “two-sided” SQL comparison expression that requires the column to be greater than or equal to the “current date/time” as of the start of the last run, and less than or equal to the current time as of the start of the current run. This extracts “everything since last time, but not including that data that changes while the extractions are running.”</p> <p>This compares SQL Server datetimes. The column used for the comparison must be declared as a datetime or some variant in SQL Server. Only the date portion of the value is used; the time portion is discarded.</p>
DATE_LAST_VALUE	High	<p>Expands to the “current date” as of the start of the last run. This expands to a value, not to an expression, allowing you to make your own expressions.</p> <p>This compares SQL Server datetimes. The column used for the comparison must be declared as a datetime or some variant in SQL Server. Only the date portion of the value is used; the time portion is discarded.</p>

Table 6: Extraction Set Identification Macros

TIMESTAMP_FILTER [column_name]	High	<p>Expands to a "one-sided" SQL comparison expression that requires the column to be greater than or equal to the current timestamp as of the start of the last run. This extracts "everything since last time."</p> <p>This compares SQL Server timestamps, which actually have no time or date information in them. The column used for the comparison must be declared as a timestamp type in SQL Server, not as a time or date.</p>
TIMESTAMP_FILTER_RANGE [column_name]	High	<p>Expands to a "two-sided" SQL comparison expression that requires the column to be greater than or equal to the current timestamp as of the start of the last run, and less than or equal to the current time as of the start of the current run. This extracts "everything since last time, but not including that data that changes while the extractions are running."</p> <p>This compares SQL Server timestamps, which actually have no time or date information in them. The column used for the comparison must be declared as a timestamp type in SQL Server, not as a time or date.</p>
TIMESTAMP_LAST_VALUE	High	<p>Expands to the "current timestamp" as of the start of the last run. This expands to a value, not to an expression, allowing you to make your own expressions.</p>

Table 6: Extraction Set Identification Macros

YYYYMMDD_FILTER [column_name]	High	This is the same as a DATE_FILTER except that only the "day" portion of the date/times is used. (Some business semantics are most meaningful when applied to "days.") This extracts "everything since the last day, including the last day."
YYYYMMDD_FILTER_RANGE [column_name]	High	This is the same as a DATE_FILTER_RANGE except that only the "day" portion of the date/times is used. (Some business semantics are most meaningful when applied to "days.") This extracts "everything since the last day, including the last day, but not including today."
YYYYMMDD_LAST_VALUE	High	Expands to the "current date in YYYYMMDD format" as of the start of the last run. This expands to a value, not to an expression, allowing you to make your own expressions.

Table 7: Smart Extraction Macros

Macro	Usage	Description
METADBNAME	Medium	Returns the name of the metadata database (EpiMeta).
MARTDBNAME	Medium	Returns the name of the datamart database (EpiMart).
INITIAL_LOAD	Medium	Expands to its argument if the initial load flag is set, or else expands to nothing.
NOT_INITIAL_LOAD	Medium	Expands to its argument if the initial load flag is set, or else expands to nothing.

Table 8: Vendor-independent Macros

Macro	Usage	Description
ADD_DAYS	Medium	Returns a date representation of its first argument plus its second argument as a number of days: For example: SELECT \$\$ADD_DAYS[\$\$DBNOW ~.~ 1]
ADD_MONTHS [date_expression ~.~ number]	Medium	Takes two arguments: adds the second as a number of months to the first argument, which is a date.
ASSERT_INDEX_EXISTS [index_name]	Low	Causes a SQL error if the index named in argument 1 does not exist. For example: \$\$BEGIN_ASSERT_INDEX \$\$ASSERT_INDEX_EXISTS[XPKCustomerMap_B] \$\$ASSERT_INDEX_EXISTS[XPKApplicationMap_B] \$\$END_ASSERT_INDEX
BEGIN_ASSERT_INDEX	Low	In Oracle, declares the DECLARE INDEX_NOT_EXISTS exception. See ASSERT_INDEX_EXISTS.
BOOL_TO_YN[testvalue]	Low	Returns the string N equals 0. Otherwise, returns 'Y'. For example: \$\$BOOL_TO_YN[6] becomes 'Y'.

Table 8: Vendor-independent Macros

CASE_BEGIN	Medium	<p>Begins a case statement that compares an expression to a list of options and returns the value for the first matching option. Also provides a single option/value pair.</p> <p>For example:</p> <pre>SELECT \$\$CASE_BEGIN[col_name ~.~ option1 ~.~ val1] \$\$CASE_ELSEIF[col_name ~.~ option2 ~.~ val2] \$\$CASE_ELSE[else_val]\$\$CASE_END</pre>
CASE_ELSE	Medium	<p>Fall-through value for a case statement that compares an expression to a list of options and returns the value for the first matching option. See CASE_BEGIN.</p>
CASE_ELSEIF	Medium	<p>Continues a case statement that compares an expression to a list of options and returns the value for the first matching option. See CASE_BEGIN.</p>
CASE_END	Medium	<p>Ends a case statement that compares an expression to a list of options and returns the value for the first matching option. See CASE_BEGIN.</p>
CAT	High	<p>Used as an operator to append "two" \$\$CAT 'strings'.</p> <p>For example:</p> <pre>\$\$TO_CHAR[table1.col1] \$\$CAT '-' \$\$CAT \$\$TO_CHAR[col2]</pre>

Table 8: Vendor-independent Macros

CBIN_VAL[testval ~, lowerbound ~, upperbound ~, binletter' CBIN_END	Medium	Can be used to "bin" numeric values into character buckets. Multiple \$\$CBIN_VAL macros should be followed by a single \$\$CBIN_END. If testval is in the range lowerbound to upperbound (inclusive), then the expression yields binletter. For example: \$\$CBIN_VAL[7 ~, 1 ~, 5 ~, 'A'] \$\$CBIN_VAL[7 ~, 6 ~, 10 ~, 'B'] \$\$CBIN_END returns the value 'B'.
COUNTER	Medium	Returns an integer column that counts the returned rows. SELECT \$\$COUNTER the_counter.
CHAR_1	Medium	Expands into a type definition for a single character field.
COUNT_ROWS_FROM COUNT_ROWS_ SELECT	Low	Can be used to count the number of rows in a table. Uses sysindexes on SQL Server for fastest count. For example: SELECT \$\$COUNT_ROWS_SELECT the_count \$\$COUNT_ROWS_FROM[MyTable]
CREATE_INDEX_IF_ NOT_EXISTS [index_type~, index_name~, table_name~, column_list~, after_creation_clause]	Low	Creates an index if it is not already there. \$\$DDL_BEGIN \$\$CREATE_INDEX_IF_NOT_EXISTS[UNIQUE ~, XPK_123 ~, table1 ~, ss_key , iss , date_key , transtype_key , seq ~,] \$\$DDL_END

Table 8: Vendor-independent Macros

DBNOW	High	<p>Returns the date/time from the database engine. For example:</p> <pre> SELECT Col1 ss_key \$\$DBNOW date_modified from zork </pre>
DDL_BEGIN	Low	<p>Starts a block of code that changes the schema. Use when there is no DECLARE statement already started. For example:</p> <pre> \$\$DDL_BEGIN \$\$NOT_DEBUG[\$\$DROP_TABLE_IF_EXISTS[table1]] \$\$DDL_END </pre>
DDL_BEGIN_NO_DECLARE	Low	<p>Starts a block of code that changes the schema. Use when there is a DECLARE statement already started. For example:</p> <pre> DECLARE \$\$VAR[transFIXED] \$\$VARCHAR_50\$\$EOS \$\$DDL_BEGIN_NO_DECLARE \$\$VAR_ASSIGN_BEGIN[transFIXED] SELECT \$\$TO_CHAR[transtype_key] \$\$VAR_ASSIGN_INTO[transFIXED] FROM Transtype_O WHERE name = FINV_ADJUST \$\$VAR_ASSIGN_END </pre>

Table 8: Vendor-independent Macros

DDL_END	Low	<p>Ends a block of SQL that changes the schema. For example:</p> <pre> \$DDL_BEGIN \$DROP_TABLE_IF_EXISTS{ \$FCTTBL[]\$NEXT] \$DROP_TABLE_IF_EXISTS{ \$FCTTBL[]_INC] \$DDL_END </pre>
DDL_EXEC [statement]	Low	<p>All items in the argument list are evaluated at runtime, not when the statement is parsed. This macro can construct SQL based on the values of variables computed in the same SQL block. For example:</p> <pre> \$DDL_BEGIN \$DDL_EXEC[CREATE INDEX X_table1 ON table1 (iss. ss_key. date_key)] \$DDL_END </pre>

Table 8: Vendor-independent Macros

DECLARE_BEGIN	Low	<p>Starts a DECLARE block. For example:</p> <pre> \$DECLARE_BEGIN \$DECLARE_BODY[\$VAR[count_INC] \$EPIINT] \$DECLARE_BODY[\$VAR[count_FC] \$EPIINT] BEGIN \$VAR_ASSIGN_BEGIN[count_INC] SELECT COUNT(1) \$VAR_ASSIGN_INTO[count_INC] FROM \$FCTTBL[]_INC \$VAR_ASSIGN_END </pre>
DECLARE_BODY [argument]	Low	Treats its argument as a declaration. See DECLARE_BEGIN.
DROP_INDEX[table_name~,~ index_name]	Medium	<p>Drops the index. For example:</p> <pre> \$DDL_BEGIN \$DROP_INDEX[table1 ~,~ index_name] \$DDL_END </pre>
DROP_TABLE_IF_EXISTS [table_name]	Medium	<p>Drops the table without returning an error indicating that the table does not exist. For example:</p> <pre> \$DDL_BEGIN \$DROP_TABLE_IF_EXISTS[table1] \$DROP_TABLE_IF_EXISTS[table2] \$DDL_END </pre>
ELSE	Medium	The start of the negative clause of an IF statement.
END_ASSERT_INDEX	Low	Ends a block of checks that indexes exist. See ASSERT_INDEX.

Table 8: Vendor-independent Macros

END_IF	Medium	Ends an IF statement. see IF.
EOS	Low	Ends a SQL statement. SELECT PROCESSED, COUNT(1), 1100 FROM table1 \$SEOS
EPIINT	Medium	Declares an int. For example: \$DECLARE_BEGIN \$DECLARE_BODY[\$VAR[unjoined] \$EPIINT] \$DECLARE_BODY[\$VAR[processed] \$EPIINT]
EPIKEY	Medium	Declares an Epiphany dimension key. See EPIINT.
FACTMONEY	Medium	Declares a monetary value. See EPIINT.
FACTQTY	Medium	Declares a decimal value. See EPIINT.
FLOAT	Medium	Declares a float value. See EPIINT.
IDENTITY	Medium	Declares a integer serial sequence. See EPIINT.
IF [condition]	Medium	Performs a conditional action. For example: \$IF[\$VAR[fc_exists] = 0] \$DDL_EXEC[\$SELECT_INTO_BEGIN[temp_table] SELECT * \$SELECT_INTO_BODY[temp_table] FROM Old_table WHERE 1=0] \$END_IF \$DDL_END

Table 8: Vendor-independent Macros

IJ_FROM [table_name1 ~,~ table_name2]	Low	Performs an inner join on two tables. See JOIN_WHERE.
INSTR[s1 ~,~ s2]	Medium	<p>Returns the position of s1 in s2. For example:</p> <p><code>\$\$INSTR[b'~,~ abc']</code></p> <p>returns 2.</p>
JOIN_LEFT_OUTER	Medium	<p>Produces a condition for outer joining the first argument to the second. For example:</p> <p><code>SELECT</code> ... <code>WHERE</code> <code>\$\$JOIN_LEFT_OUTER[t1.c1 ~,~ t2.c2]</code></p>
JOIN_RIGHT_OUTER	Medium	<p>Produces an equals sign appropriate for right outer joins (No arguments are needed.) For example:</p> <p><code>SELECT</code> ... <code>WHERE</code> <code>t1.c1 \$\$JOIN_RIGHT_OUTER t2.c2</code></p>

Table 8: Vendor-independent Macros

JOIN_WHERE [join_condition]	Low	<p>Supplies the WHERE clause for a join. For example:</p> <pre> SELECT col1, col2 FROM Table1 s \$\$LOJ_FROM[table2 m ~.~ s.iss = m.iss AND s.col2=m.col2] \$\$LOJ_FROM[table3 d ~.~ m.col1 = d.col1] WHERE 1=1 \$\$JOIN_WHERE[m.col1=d.col1(+)] \$\$JOIN_WHERE[s.iss = m.iss (+) AND s.col2 = m.col2 (+)] </pre>
LENGTH[s]	Medium	<p>Returns the length of a string. For example:</p> <pre> \$\$LENGTH['abc'] </pre> <p>returns 3.</p>
LOJ_FROM [join_condition]	Low	Performs a left outer join. See JOIN_WHERE.
MAX_SYS_DATE	Medium	Returns the highest date supported by the database.
MODULO{x ~.~y}	Low	<p>Returns the remainder when x is divided by y. For example:</p> <pre> MODULO{7 ~.~ 4} </pre> <p>returns 3.</p>

Table 8: Vendor-independent Macros

NBIN_VAL[testval ~,~ lowerbound ~,~ upperbound ~,~ binnumber] NBIN_END	Medium	Can be used to “bin” numeric values into numeric buckets. Multiple \$\$NBIN_VAL macros should be followed by a single \$\$NBIN_END. If testval is in the range lowerbound to upperbound (inclusive), then the expression yields binnumber. For example: \$\$NBIN_VAL[7 ~,~ 1 ~,~ 5 ~,~ 1] \$\$NBIN_VAL[7 ~,~ 6 ~,~ 10 ~,~ 2] \$\$NBIN_END return the value 2.
NO_FROM_LIST	Medium	Supplies the “dummy” FROM clause needed by some database vendors. For example: SELECT 'MODIFIED'. \$\$VAR[modified]. 1050 \$\$NO_FROM_LIST\$\$EOS
NUMBER(9)	Medium	Declares a decimal(9). See EPIINT.
NUMBER(9.2)	Medium	Declares a decimal(9.2). See EPIINT.
NUMBER(9.5)	Medium	Declares a decimal(9.5). See EPIINT.
NVL [expression ~,~ value]	High	When the first argument is NULL, replace it with the value in the second argument. For example: SELECT \$\$TO_CHAR[\$\$NVL[MAX(col1) ~,~ 1]]
ORACLE [expression]	High	Expands to nothing if the database is not Oracle. For example: SELECT COUNT(1) FROM \$\$\$SQLSERVER[sysobjects]\$\$ORACLE[tabs]

Table 8: Vendor-independent Macros

RAISE_EXCEPTION [Exception]	Low	Raises the given exception (as a variable on Oracle, as a string on SQL Server). For example: \$\$RAISE_EXCEPTION[MyException] raises an exception.
RENAME_OBJECT	Medium	Renamed tables or other database objects. For example: \$\$RENAME_OBJECT[oldtablename ~.~ newtblname]
SELECT_INTO_BEGIN [table_name]	High	Creates a table from a SELECT statement. Expands into a CREATE TABLE AS or a SELECT INTO statement. For example: \$\$SELECT_INTO_BEGIN[temp_tab] SELECT * \$\$SELECT_INTO_BODY[temp_tab] FROM Old_tab WHERE 1=0
SELECT_INTO_BODY [table_name]	High	Creates a table from a SELECT statement. Expands into a CREATE TABLE AS or a SELECT INTO statement. See SELECT_INTO_BEGIN.
SMALLDATE	Medium	Declares a SMALLDATETIME. See EPIINT.
SMALLINT	Medium	Declares a double-byte integer. See EPIINT.
SQLSERVER [expression]	High	Expands into nothing if the database engine is not SQL Server. For example: SELECT COUNT(1) FROM \$\$\$SQLSERVER[sysobjects]\$\$ORACLE[tabs]

Table 8: Vendor-independent Macros

SSKEY	Low	Declares the type Epiphany uses for ss_keys. See EPIINT.
SUBSTRING [expression ~,~ start ~,~ length]	Medium	Performs a substring operation. For example: \$\$\$SUBSTRING[name ~,~1 ~,~8]
SUPERNVL	Medium	Converts a null value for the first argument into the second argument. The resulting column is NOT-NULL able in the schema definition of the result set. For example: SELECT \$\$\$SUPERNVL[col1 ~,~ 'UNKNOWN']
TABLE_EXISTS_CONDITION [table_name]	Low	Detects if a table exists. For example: SELECT COUNT(1) FROM \$\$\$SQLSERVER[sysobjects]\$ \$ORACLE[tabs] WHERE \$\$TABLE_EXISTS_CONDITION[table_name]
TABLE_WITH_PREFIX [database_name ~,~ table_name]	Medium	Qualifies a table name with a database or user name. For example: SELECT source_system_key iss FROM \$\$TABLE_WITH_PREFIX[\$\$METADBNAME ~,~ source_system]
TINYINT	Medium	Declares a single-byte integer. See EPIINT.
TO_CHAR [expression]	High	Converts a value to a character. Length is second argument. For example: SELECT \$\$TO_CHAR[\$\$NVL[MAX(col1) ~,~ 1]]

Table 8: Vendor-independent Macros

TO_DATE [expression]	Medium	Converts a value to a database date.
TO_DATEFMT [expr ~,~ format]	Low	Converts expression to a date type with the appropriate Oracle format (<i>format</i> is ignored on SQL Server).
TO_EPIDATE [expression]	High	Converts a date to the string format preferred by EpiChannel. For example: Select col1 ss_key. \$\$TO_EPIDATE[date_col] date_modified from zork
TO_NUMBER	Medium	Converts an expression to a number. For example: \$\$TO_NUMBER['123'] returns 123.
TO_TIME	Medium	Converts its argument to a time representation. For example: SELECT \$\$TO_TIME[\$\$DBNOW]
TO_YYYYMMDD [expression]	Medium	Converts a data to a YYYYMMDD string.

Table 8: Vendor-independent Macros

TRANSLATE_BEGIN TRANSLATE_VAL [expression ~,~ searchval ~,~ translationval ~,~ OtherTerm TRANSLATE_ELSE TRANSLATE_END	Medium	Searches expression for occurrences of each searchval and returns the translationval. Note that \$\$TRANSLATE_VAL terms should be nested inside of each other. An optional \$\$TRANSLATE_ELSE can be nested inside the final \$\$TRANSLATE_VAL For example: \$\$TRANSLATE_BEGIN \$\$TRANSLATE_VAL['abcdef' ~,~ bce ~,~ 'Value1' ~,~ \$\$TRANSLATE_VAL['abcdef' ~,~ cde ~,~ 'Value2' ~,~ \$\$TRANSLATE_ELSE['Other']]] \$\$TRANSLATE_END returns 'Value2'
VAR [variable_name]	Medium	References a database variable. For example: SELECT PROCESSED', \$\$VAR[processed]. 1100 \$\$NO_FROM_LIST\$\$EOS
VAR_ASSIGN_BEGIN [variable_name]	Medium	Assigns to a database variable. For example: \$\$VAR_ASSIGN_BEGIN[max_key] SELECT \$\$TO_CHAR[\$\$NVL[MAX(col1) ~,~ 1]] \$\$VAR_ASSIGN_INT0[max_key] FROM table2 \$\$VAR_ASSIGN_END.
VAR_ASSIGN_END	Medium	Assigns to a database variable. See VAR_ASSIGN_BEGIN.
VAR_ASSIGN_INT0 [variable_name]	Medium	Assigns to a database variable. See VAR_ASSIGN_BEGIN.
VARCHAR_100	Medium	Declares a variable-width character datatype that holds a maximum of 100 characters. See EPIINT.

Table 8: Vendor-independent Macros

VARCHAR_15	Medium	Declares a variable-width character datatype that holds a maximum of 15 characters. See EPIINT.
VARCHAR_25	Medium	Declares a variable-width character datatype that holds a maximum of 25 characters. See EPIINT.
VARCHAR_255	Medium	Declares a variable-width character datatype that holds a maximum of 255 characters. See EPIINT.
VARCHAR_5	Medium	Declares a variable-width character datatype that holds a maximum of 5 characters. See EPIINT.
VARCHAR_50	Medium	Declares a variable-width character datatype that holds a maximum of 50 characters. See EPIINT.

Table 9: Testing Macros

Macro	Usage	Description
DEBUG	Low	<p>If debugging is enabled: Expands to nothing if the extract command's verbosity level is less than 3; otherwise, returns its argument.</p> <p>Note: For testing that depends on whether debugging is on or off, use both DEBUG and NOT_DEBUG.</p>
NOT_DEBUG	Low	<p>If debugging is NOT enabled: Expands to nothing if the extract command's verbosity level is higher than 3; otherwise, returns its argument.</p> <p>Note: For testing that depends on whether debugging is on or off, use both DEBUG and NOT_DEBUG.</p>

APPENDIX B

EPICENTER CONFIGURATION

The Configuration dialog box shows the various configurations for your EpiCenter.

To view or modify configuration settings, choose Configuration from the EpiCenter menu. The Configuration dialog box (see Figure 78) is displayed. It has five tabs: General, Transaction Types, Measure Units, Option Labels, and Ticksheet Types.

The Configuration dialog box is the user interface for the *config_master* metadata table that contains various system parameters in name/value format.

GENERAL SETTINGS

The configuration data that appears in the General tab of the Configuration dialog box has been set for a default EpiCenter. Other than entering your e-mail password, the information should be correct, or require minimal alteration.

You may modify these settings if necessary. Select the key in the list and enter a new value in the Value textbox, and click Update. All of the values are defined in the dialog box.

Note the following:

- You need to change the Mail Password after installing the Epiphany software.
- The Mail Profile name is the default e-mail address for the EpiCenter Manager user.

General Settings

- **current_datamart A / B**
Indicates which set of tables (A or B) is active. See “Mirroring: A and B Tables” on page 83.
- **date_type**
Calendar is the default.

date_445 represents the 13-week-per-quarter calendar in which the months in the quarter are defined as consisting of 4 weeks, 4 weeks, and 5 weeks.
- **end_year**
The ending year of the EpiMart. The date range of the EpiMart should be at least as large as any dates that are found in the data you will be extracting.

In order for users to get the best results when forecasting trends using Relevance, you need to build the date dimension as far into the future as you plan to forecast. (Currently, the maximum prediction is three years past the last date that has recorded data.) If the date dimension is not built out far enough, the user will receive columns with names such as
Dec 1999, Second Next, Third Next instead of *Dec 1999, Dec 2000, Dec 2001*.
- **fiscal_year start_m[onth]**
If the chosen month is other than January, then the actual starting year is one less than the start_year value (thus the given year may start on January 1 and still be a complete fiscal year).
- **number_of_years**
The number of years during which the data warehouse will be operational. The default is 20.
- **min_sample_invlog10**
The minimum compression ratios for Momentum sampling. If set to zero, then no sampling occurs. If set to 2, then 1/100 of the sampling occurs. The value 3 means 1/1000th, and the value 1 means 1/10th of the sampling occurs, and so forth.

- mom_inlist_limit

The maximum number of elements that your system allows in the list used by a SQL IN clause. By default this value is 254 on Oracle and 2,000 on SQL Server.

For example, for a query of the type:

```
SELECT *FROM table WHERE attr#i in list of values
```

mom_inlist_limit specifies the maximum number of values that can be placed in the *list of values* in the above query.

Note: Exporting or importing deletes the value set for mom_inlist_limit. If you export metadata from a SQL Server EpiCenter to an Oracle metadata (or vice versa), you need to re-enter this value.

- start_day_445

The beginning day of the 445 quarter.

- start_year

The year that the EpiMart becomes operational. The default is 1990.

- version

The version number of this EpiCenter Manager.

- week_start_day

The day of the week that is the first day of the week. Sunday is the default.

- last_extract_date

The date that appears as the *Data is valid as of...* in Clarity reports.

Transaction Types

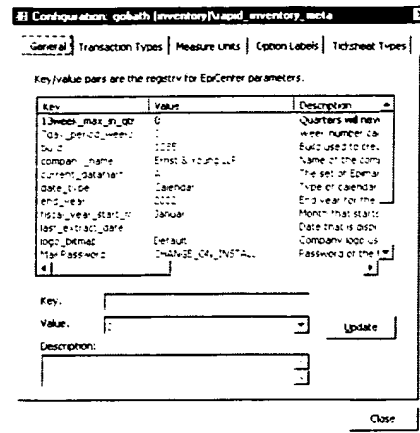


Figure 78: Configuration Dialog Box: General Tab

TRANSACTION TYPES

The current transaction types are shown in the Transaction Types tab (Figure 79) of your Configuration dialog box. These are the typical transaction types for a generic installation. Your site may need additional transaction types for complicated measures.

Use this tab to customize the transaction types for your site. Keys 1-99 are reserved for booking transaction types. Keys 101-199 are reserved for shipping transaction types.

After modifying the transaction types, click Update, which writes the new data to the EpiMeta. Use the Propagate button to synchronize the *transtype_0* table in the EpiMeta and EpiMart databases.

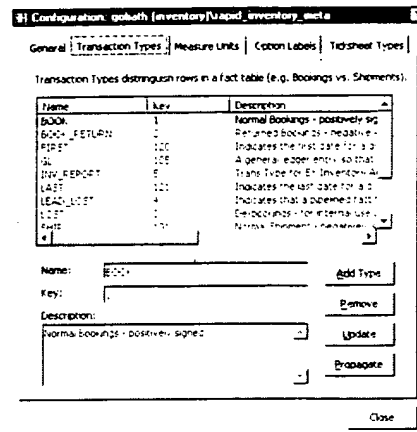


Figure 79: Configuration Dialog Box: Transaction Types

MEASURE UNITS

Measure units define how currency units, percentages, and numerical units are displayed by default on ticksheets. To define a measure unit, enter its name and symbol, such as the dollar or percent sign. The symbol character can be at most one character. To input the Yen sign and other similar characters, use the Windows character map to copy and paste the symbol.

For a currency unit, if this EpiCenter uses currencies from different countries, select the Multi-currency option. Multi-currency controls how the system handles currencies. (A GROUP BY is forced on the currency dimension so that the system does not calculate different currencies, such as French francs and British pounds, as the same currency.) Also, select Postfix if the symbol should follow the number: 100% is postfix, whereas \$100 (the default) is prefix.

Option Labels

Add any description for your reference in the Description textbox, and click Add Unit. The system generates a hidden key for the unit. EpiCenter Manager uses this key when flagging a measure with a unit designation.

Configuration: goliath_inventory\typed_inventory_msta

General | Transaction Types | Measure Units | Option Labels | Ticksheet Types

Measure units control the display of numeric quantities

Name	Symbol	Multi-Currency	Post
CURRENCY		0	0
CURRENCY_LOCAL		0	0
CURRENCY_USD	\$	0	0
PERCENT	%	0	0
UNIT		0	0

4 | 1

Name: CURRENCY Add Unit

Symbol: ☐ Multi-Currency ☐ Postive Remove

Description: Default Currency Unit - with \$ symbol Update

Close

Figure 80: Configuration Dialog Box: Measure Units Tab

OPTION LABELS

The Option Labels tab lists the labels for the option names as they appear to end users on ticksheets. Although the option names are set by the Epiphany system, you can customize option labels for your ticksheets. For example, you could change the option label for the option name *UnitQuantity* to *Quantities*. To do this, select the Option Name in the upper panel of the tab, enter the name you want to appear on the ticksheet in the Option Label textbox, and click Update.

The Options Labels tab also contains names and values that need to be configured for Momentum ticksheets.

MOMENTUM LABELS

As part of configuring Momentum, you need to modify the default option labels and default value names and labels. Use the upper panel of the Options Labels tab to modify option labels, and the lower panel of the tab to modify value names and labels for the selected option name.

The following option names are specific to Momentum:

- **HOUSEHOLD_CARDINALITY**
Corresponds to the drop-down list that appears on top of the filter pop-up window when one creates a list of households. The value names and default labels are as follows:
 - 0 = For household (select when filtering on household information).
 - 1 = No one in household (select when filtering on individual information—no one).
 - 2 = No more than one in household (select when filtering on individual information—no more than one).
 - 3 = Exactly one in household (select when filtering on individual information—exactly one).
 - 4 = At least one in household (select when filtering on individual information—at least one).
 - 5 = Everyone in household (select when filtering on individual information—everyone).
- **INDIVIDUAL_CARDINALITY**
Corresponds to the label on the top of the filter pop-up window when creating a list of individuals. The value names and default labels are as follows:
 - 0 = No cardinality (you can change this to something such as “Individual filter”).
- **MOMENTUM_HOUSEHOLD_LABELS**
Miscellaneous labels used on the Momentum household filter pop-up window. The value names and default labels are as follows:
 - filter_title: Filter Title (title on top of household filter pop-up window).

Momentum Labels

- `filter_description`: Filter Description (text displayed directly under the title in the filter pop-up window).
- `transaction_description`: Transaction Filter Description (text displayed above transaction drop-down list in the filter pop-up window).
- `no_transaction`: None (text in the transaction drop-down list indicating that no transaction has been selected).
- **MOMENTUM_INDIVIDUAL_LABELS**
Miscellaneous labels used on Momentum individual filter pop-up windows. The value names and default labels are as follows:
 - `filter_title`: Filter Title (title on top of individual filter pop-up window).
 - `filter_description`: Filter Description (text displayed directly under the title in the filter pop-up window).
 - `transaction_description`: Transaction Filter Description (text displayed above the transaction drop-down list in the filter pop-up window).
 - `no_transaction`: None (text in the transaction drop-down list indicating that no transaction has been selected).
- **MOMENTUM_MEASURE_MODE**
Defines the labels next to the radio buttons to the right of the transaction drop-down list in the filter pop-up window. These radio buttons determine if the transaction filter should be positive or negative (should match or should not match). The value names and default labels are as follows:
 - 0 = Nothing (label next to radio button for negative case).

- 1 = Some thing (label next to radio button for positive case).

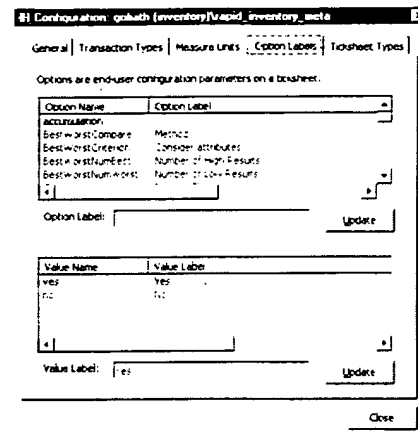


Figure 81: Configuration Dialog Box: Option Labels Tab

Note: When configuring Momentum for an EpiCenter, you can use the Ticksheet Types tab of the Configuration dialog box (Figure 82) to change the ticksheet types labels from their default values, which are Filter Individuals and Filter Households. These are the labels that display in the main menu of the browser window (see Figure 28, on page 129).

TICKSHEET TYPES

The Ticksheet Types tab is primarily read-only and shows the contents of the various ticksheet types. You can update the ticksheet type label, which is the name of the ticksheet types that display in the main menu of the browser window (see Figure 28, on page 129). When working with Momentum and configuring the terminology for Individual and Group, you need to change these labels so that the wording is correct for your installation.

This tab is the means by which you disable an entire application by choosing the application, such as Momentum, and deselecting Enabled. This is useful if you want to test Clarity and Relevance ticksheets, but have not yet set up Momentum.

Ticksheet Types

Configuration: g:\path\inventory\maped_inventory_meta

General | Transaction Types | Measure Units | Option Labels | **Ticksheet Types**

Each distinct Epiphany module is defined by a ticksheet type.

Application: ☒ Enabled

Ticksheet Type	Label	Template	Cl
clarity	Table	clarityticksheet	co

Type Label:

Attribute Role	Option Name
ClarityColumn	UnitLabel2
ClarityRow	UnitQuantity
	Sum
	Percents
	Precision
	Charts
	Flags Rows
	Max Col

Figure 82: Configuration Dialog Box: Ticksheet Types Tab

APPENDIX C

DATE DIMENSION FIELDS

The date dimension fields used by the Epiphany system are described below.

Table 10: Date Dimension Fields

dim_col_name	Description
cq_and_cy_name	Calendar quarter and year name. For example, Q1 1999.
cq_name	Calendar quarter name. For example, Q1.
cy_name	Calendar year name. For example, 1999.
date_key	Primary key—date as a native date type.
day_cq_begin	Whether or not this is a day on which a calendar quarter begins. (1/0).
day_cq_end	Whether or not this is a day on which a calendar quarter ends. (1/0).
day_cy_begin	Whether or not this is a day on which a calendar year begins. (1/0).
day_cy_end	Whether or not this is a day on which a calendar year ends. (1/0).

Date Dimension Fields

Table 10: Date Dimension Fields

day_fq_begin	Whether or not this is a day on which a fiscal quarter begins. (1/0).
day_fq_end	Whether or not this is a day on which a fiscal quarter ends. (1/0).
day_fy_begin	Whether or not this is a day on which a fiscal year begins. (1/0).
day_fy_end	Whether or not this is a day on which a fiscal year ends. (1/0).
day_month_begin	Whether or not this is a day on which a month or period begins. (1/0).
day_month_end	Whether or not this is a day on which a month or period ends. (1/0).
day_name	The day as a native date type.
day_name_char	Date (as a string). For example, Apr 02 1995.
day_name_char_weekday	Date (as a string) with weekday prefix. For example, Sun Apr 02 1995.
day_number_in_cq	The number of this day in the calendar quarter, starting at 1.
day_number_in_cy	The number of this day in the calendar year, starting at 1.
day_number_in_fq	The number of this day in the fiscal quarter, starting at 1.
day_number_in_fy	The number of this day in the fiscal year, starting at 1.
day_number_in_month	The number of this day in the month or period, starting at 1.
day_number_in_week	The number of this day in the week, starting at 1.
day_number_til_end_cq	The number of days until the end of the calendar quarter, ending with 1.
day_number_til_end_fq	The number of days until the end of the fiscal quarter, ending with 1.
fq_and_fy_name	Fiscal quarter and year name. For example, Q1 1999.

Table 10: Date Dimension Fields

fq_name	Fiscal quarter name. For example, Q1.
fy_name	Fiscal year name. For example, 1999.
month_and_cy_name	The name of the month abbreviated and the calendar year. For example, Apr 1999.
month_and_fy_name	Month (abbreviated) or period name and fiscal year. For example, Apr 1999.
month_name	The three-letter abbreviations of the month (without a year or any other value). For example, Apr.
month_number	Month or period number since the first date in the system, starting at 1.
month_number_in_cq	Month number since the start of the calendar quarter, starting at 1.
month_number_in_cy	Month number since the start of the calendar year, starting at 1.
month_number_in_fq	Month or period number since the start of the fiscal quarter, starting at 1.
month_number_in_fy	Month or period number since the start of the fiscal year, starting at 1.
month_number_til_end_cy	Number of months or periods until the end of the calendar year, ending with 1.
month_number_til_end_fy	Number of months or periods until the end of the fiscal year, ending with 1.
vacation_day	Whether or not this day is a vacation, (1/0).
week_friday	Date (as a string) for the Friday of the current week. For example, Apr 01 1994.
week_monday	Date (as a string) for the Monday of the current week. For example, Apr 01 1996.
week_number	Week number since the first date in the system, starting at 1.

Date Dimension Fields

Table 10: Date Dimension Fields

week_number_cq	Week number since the start of the calendar quarter, starting at 1.
week_number_cy	Week number since the start of the calendar year, starting at 1.
week_number_fq	Week number since the start of the fiscal quarter, starting at 1.
week_number_fy	Week number since the start of the fiscal year, starting at 1.
week_number_til_end_cq	Week number until the end of the calendar quarter, ending with 1.
week_number_til_end_cy	Number of weeks until the end of the calendar year, ending with 1.
week_number_til_end_fq	Week number until the end of the fiscal quarter, ending with 1.
week_number_til_end_fy	Number of weeks until the end of the fiscal year, ending with 1.
week_saturday	Date (as a string) for the Saturday of the current week. For example, Apr 01 1995.
week_sunday	Date (as a string) for the Sunday of the current week. For example, Apr 02 1995.
weekday	Weekday prefix: for example, Sun.
month_name	The three-letter abbreviations of the month (without any year or any other value).

APPENDIX D

PHYSICAL TYPE VALUES

The tables in this appendix define the database type translations for the physical types you select in EpiCenter Manager's Base Dimension and External Tables dialog boxes. The physical type you select is replaced by its associated database type. The translation of physical type to database type depends on your RDBMS platform.

Notes to sites that use their own database types and SQL macros:

The various fact semantic types (such as Transactional/Statelike) use SUM() operators over the fact columns. New tables are created via SELECT INTO from the result.

With decimal data types such as NUMBER(16,4) that logically map to a NUMERIC(16,4) SQL Server datatype, repeated SUM operations within SQL Server may cause an automatic increase in the column precision, ultimately resulting in a NUMERIC(38) field, which occupies 17 bytes per record. For this reason, both FACTMONEY and FACTQTY map to the MONEY database type in SQL Server. This database type occupies only 8 bytes per record.

Physical Type Values

Table 11: SQL Server Physical Type Values

Physical Type	Database Type
CHAR_1	CHAR(1)
EPIINT	INT
EPIKEY	INT
FACTMONEY	MONEY
FACTQTY	MONEY
IDENTITY	INT IDENTITY
SMALLDATE	SMALLDATETIME
SMALLINT	SMALLINT
SSKEY	VARCHAR(50)
TINYINT	TINYINT
VARCHAR_5	VARCHAR(5)
VARCHAR_15	VARCHAR(15)
VARCHAR_25	VARCHAR(25)
VARCHAR_50	VARCHAR(50)
VARCHAR_100	VARCHAR(100)
VARCHAR_255	VARCHAR(255)

Table 12: Oracle Physical Type Values

Physical Type	Database Type
CHAR_1	CHAR(1)
EPIINT	NUMBER(10)
EPIKEY	NUMBER(9)
FACTMONEY	NUMBER(19,4)
FACTQTY	NUMBER(19,4)
IDENTITY	NUMBER(10)
SMALLDATE	DATE
SMALLINT	SMALLINT
SSKEY	VARCHAR2(50)
TINYINT	SMALLINT
VARCHAR_5	VARCHAR2(5)
VARCHAR_15	VARCHAR2(15)
VARCHAR_25	VARCHAR2(25)
VARCHAR_50	VARCHAR2(50)
VARCHAR_100	VARCHAR2(100)
VARCHAR_255	VARCHAR2(255)

Physical Type Values

APPENDIX E

WRITING STAGING SQL STATEMENTS

The Epiphany extraction process (excluding aggregation) consists of two phases:

- 1) loading data from source systems into the Epiphany staging tables, and
- 2) running semantic instances against these staging tables. This appendix describes the recommended practices for writing SQL statements that populate the dimension and fact staging tables.

BASE DIMENSION STAGING SQL STATEMENTS

Base dimensions describe the physical dimension tables in EpiMart. You can use EpiCenter Manager to configure the dimension columns for each base dimension table. During an extraction job, one or more SQL statements can be executed against the base dimension staging table to populate these columns.

The purpose of the base dimension staging query is to extract the latest values from the source system. You need not be concerned with determining when a value has changed because Epiphany's semantic templates perform this task.

Base Dimension Staging SQL Statements

You can use the Template feature in the SQL Statement dialog box (see Figure 52. on page 185) to provide an SQL template for a given dimension table. For example, assume you define a base dimension called Customer with the following dimension columns:

- FullName
- Age
- City
- State
- ZipCode

In the Tables References panel of the SQL Statement dialog box, select the option *Populates a dimension*. Choose the dimension table (Customer) from the drop-down list. Clicking the Template button will display the following SQL in the dialog box:

```
SELECT
    <YOUR EXPRESSION> customer_sskey,
    <YOUR EXPRESSION> data_modified,
    <YOUR EXPRESSION> FullName,
    <YOUR EXPRESSION> Age,
    <YOUR EXPRESSION> City,
    <YOUR EXPRESSION> State,
    <YOUR EXPRESSION> ZipCode
FROM
    <YOUR TABLE>
```

This is a regular SQL statement for which you must provide the values **<YOUR EXPRESSION>** and **<YOUR TABLE>**. You must assign each column in the SELECT list a valid SQL expression for the value of its destination column. A FROM clause is required to make this a valid statement; a WHERE clause is optional.

The first two columns, *customer_sskey* and *date_modified*, were not specified in the definition of the Customer base dimension. These columns are implicitly added to the base dimension table by Epiphany's Adaptive Schema Generator and must be populated by the staging SQL. The first column is a source system key (*sskey*) and should be unique for every row in the staging table. The concept of *sskey* is important in EpiMart because it tells the semantic templates whenever a row in the staging table represents the same source system entity as a row that already exists in the dimension table. The *sskey* is a variable length string, normally of maximum length 50. It corresponds to the primary key of the source system table or the tables that make up this query.

Note: If the *sskey* column is of interest to end users for querying, then a dimension column populated with the same values will need to be created because the *sskey* is not available for ticksheet configuration.

The other implicit column, *date_modified*, has the same name in all base dimension staging tables and is used to identify when a base dimension row is inserted into the EpiMart. If the source system contains a *creation date* field, then this field should be used. Otherwise, you can use the source system's expression for "right now," which causes newly extracted rows to assume the date when they were extracted into EpiMart; for Microsoft SQL Server this expression is `GetDate()`. For best results, dates should be returned as strings, for example, 5/26/1999.

The remainder of the columns in the SELECT list must be populated with an appropriate expression for the meaning of that column. Any SQL expression that can be executed against the source RDBMS is valid. Also, Epiphany provides a set of SQL macros that will be automatically expanded to the correct syntax for your source system. Use of macros facilitates the cross-platform usage of your SQL statements. See Appendix A, "Epiphany Macros" for more information.

Duplicate sskey's

An important point about these expressions is that null values are *not* allowed in any field of the staging table. The reason for this is simple: the Epiphany system uses GROUP BY statements at end-user query time to form the tables and charts of front-end applications such as Clarity. However, fact rows that aggregate on columns with null values are left out of the resulting reports because nulls are removed from GROUP BY's. Rather than incurring the query-time penalty for this check, EpiMart insists on non-null dimension column values.

To circumvent this problem, substitute the string UNKNOWN for any null values using the NVL macro. The Epiphany system will automatically generate an UNKNOWN row in your dimension table. The UNKNOWN value is configurable: if UNKNOWN is a valid value in your dimension table, use another value.

Duplicate sskey's

If during a single extraction, a staging table is loaded with two or more rows with the same *sskey*, then the last row entered is used. See Appendix F. "Semantic Types" for a description of the dimension semantic types.

Dimension Staging Queries with Joins

The Epiphany system allows the use of joins in base dimension staging queries. Star schemas typically de-normalize data structures in transactional systems into flat hierarchies, and you must be aware of what the granularity of a base dimension represents in this circumstance.

For example, you will rarely want to use a Cartesian product of two tables in a base dimension staging query, unless the *sskey* of the result set will combine the primary keys of the two tables that are being crossed. It is more common for a single table to "drive" the result set, with other tables joined through unique key lookups to provide additional textual values. For instance, a Product Master table in the source system might represent the driving table of a Product base dimension (with the *sskey* taken from the primary key of the Product Master table), but other tables with textual values for Product Line or Platform may be joined with this master table. In this case, you should ensure that the joined columns of the lookup tables are properly indexed (usually with UNIQUE indexes).

Constructing Base Dimension Queries with DISTINCT Fact Values

Sometimes dimensions are created for which no corresponding master table exists in the source system. For instance, an Order fact may have an Order type associated with it (with several possible choices). These values will be embedded directly in the fact rows on the source, but no lookup table exists with all the choices. In this case, a `SELECT DISTINCT` query against the source system's fact table might be appropriate for populating base dimension staging tables in EpiMart. The alternative to this method is the use of degenerate dimensions in the fact table, although degenerate dimensions cannot be aggregated.

FACT STAGING SQL STATEMENTS

SQL statements that populate fact staging tables are generally more complex than the ones used to load dimension staging tables. As with base dimension tables, the columns of the `SELECT` statements are determined by the metadata definition of the fact table (and its constellation) along with certain implicit rules.

To illustrate this point, assume that you define an Order fact in a Sales constellation with the following dimension roles:

- CustomerBillTo
- Product
- CustomerShipTo
- SalesPerson

The constellation contains a single degenerate dimension called OrderNumber, and the Order table has two fact columns: *net_price* and *number_units* that represent the extended amount for an order line item, along with the quantity.

Clicking the Template button on the SQL Statement dialog box (with the *Populates fact table* option selected and the Order table selected in the drop-down list) displays the following SQL in the dialog box:

Fact Staging SQL Statements

```
SELECT
    <YOUR EXPRESSION> ss_key,
    <YOUR EXPRESSION> date_key,
    <YOUR EXPRESSION> transtype_key,
    <YOUR EXPRESSION> process_key,
    <YOUR EXPRESSION> customerbillto_sskey,
    <YOUR EXPRESSION> product_sskey,
    <YOUR EXPRESSION> customershipto_sskey,
    <YOUR EXPRESSION> salesperson_sskey,
    <YOUR EXPRESSION> ordernumber_key,
    <YOUR EXPRESSION> net_price,
    <YOUR EXPRESSION> number_units
FROM
    <YOUR TABLE>
```

As with base dimension staging queries, you must identify what a row in this fact table represents. Based on the columns in this example, a row seems to indicate a line item of a sales order. (In this case, the assumption is that the salesperson gets full credit for a line item; another interpretation of this fact row might be a particular amount of credit that a salesperson received for an order line item.) Typically, the FROM clause of this query would join the Order Line Item table to the Order Header table in the source system.

The columns in the SELECT list can now be divided into these categories:

- Implicit columns that were added automatically
- Dimension role foreign keys
- Degenerate dimension keys
- Fact numeric columns

First, consider implicit columns. As with base dimensions, each fact staging row contains an *ss_key* (notice the difference in spelling) that uniquely identifies this row in the source system. In this example, the *ss_key* might be a concatenation of the Order Number with the Order Line Number (since this combination is presumably unique). *ss_key*'s will be used on subsequent extractions to prevent duplicate copies of the fact row from being created in EpiMart.

The *date_key* indicates when the fact occurred. Since time is a central component of EpiMart, each fact table must contain this column. Many facts are time based; in this example, *date_key* represents the time when the order was placed. However, if time is not important for this fact, then the current system time can be used as a placeholder. Note that *date_key* is granular only to that single *day* when the fact occurred. For best results, the fact SQL Statement should return the day as a string, for instance, 5/1/1999.

Transaction type is another central concept of fact table processing in EpiMart. The SQL statement should return a numeric key that matches with one of the transaction types defined on the Configuration dialog box in EpiCenter Manager. (See Appendix F, "Semantic Types" for more information about *transtype_key*.)

The *process_key* identifies rows from the fact table to be processed by a specific semantic type. (A fact table can contain different types of rows, requiring different semantic types.) For example, the Order fact might hold both Bookings and Shippings, and *process_key* would identify which fact staging rows were which. (See Appendix E, "Writing Staging SQL Statements" for more information about *process_key*.)

Next, you must enter values for each of the dimension role foreign keys. Notice that the names of the columns in the SQL template are *DimRoleName_sskey*. These fields refer back to *sskey*'s of the base dimension tables for this fact. You need to understand the meaning of each base dimension table to ensure that the keys resolve properly. If the *sskey* of the Product base dimension is taken from a Product Master list in the source system, then *product_sskey* in the Order table must also refer to an entry in the Product Master. If a base dimension is the cross-product of two source system tables, the fact staging keys for that dimension must also represent a unique cross-product entry.

Using External Tables as Inputs to Staging Queries

Degenerate keys in the fact staging query should be populated with string values. In the example above, the *ordernumber_key* field would probably be populated with the actual primary key of the Order Header table, such as Order Number 253AD56.

Finally, the numeric columns represent the actual quantities and raw amounts that are associated with each fact entry. Each column should be an additive amount for correct front-end query results. For instance, total dollar amounts for a line item should be populated instead of unit prices because unit prices cannot be added across fact rows.

USING EXTERNAL TABLES AS INPUTS TO STAGING QUERIES

Sometimes it may be necessary to bring data into EpiMart external (temporary) tables before performing any joins; for example, if the source system's SQL limits your ability to manipulate the data. The full power of EpiMart's RDBMS engine can then be used to load the staging tables. In this case, the following sequence of actions is usually employed:

- Step 1:** Drop any indexes on the external tables for fast loading.
- Step 2:** Load the external tables from the source system.
- Step 3:** Create any indexes on external tables needed for fast joins.
- Step 4:** Load the staging tables using queries against the EpiMart External tables. All query plans should use the indexes built in Step 3.

APPENDIX F

SEMANTIC TYPES

This appendix describes the dimension and fact semantic types.

Note: Fact rows with all zero facts are discarded by all semantics.

DIMENSION SEMANTIC TYPES

The dimension semantic types are Slowly Changing Dimensions, Latest Dimension Value, First Dimension Value, and Initial Dimension Load.

Slowly Changing Dimensions

A Slowly Changing Dimension is a dimension in which the attributes or hierarchy of the dimension can change over time, but historical data is not restated. Two examples follow:

- A sporting goods chain decides to rename a store. By using the Slowly Changing Dimensions semantic type, the old name is retained for all of the historical data. One can still identify when the store changed names and compare sales before and after the name change.

Slowly Changing Dimensions

- This same chain is national and has stores divided into three regions. On January 1, 1998, the chain reorganizes its regions, moving Denver from the Central to the Western region. Their 1998 sales forecasts take into account that the Denver store is in the Western region, but they do not want to recalculate forecasts and actuals from previous years. Using the Slowly Changing Dimensions semantic type, sales for Denver can be aggregated up to the Central region through 1997. Beginning January 1, 1998, Denver sales are applied to the Western Region.

The Slowly Changing Dimensions semantic type accomplishes the following logic:

- Rows with the same *sskey* in the staging table will be eliminated following in a “last in wins” rule. This is determined by the special column called *key*, which is created by EpiChannel and is automatically incremented during normal extractions. The highest *key* row for a given *sskey* will be accepted.
- New rows are created by searching through the dimension staging table for new *sskey* values, or *sskey* values that have one or more dimension column changes from the last known values. Each of these cases creates a new row in the dimension table. The mapping row for that *sskey* points to the latest dimension row with that *sskey* value.
- In the EpiChannel log file, new *sskey* rows are reported in the *Inserted* column. The number of changed rows is reported in the *Modified* column. The number of rows in the staging table is reported in the *Processed* column.
- The dimension table has its key column made into a Primary Key index. Each dimension column is also Indexed (non-uniquely). The Mapping table is indexed (primary key) on *iss*, *sskey*. (You can use the Dimension Column dialog box in EpiCenter Manager to disable the indexing of individual columns.)

iss is used when two source systems have the same *sskey* values (such as when the SAP and Vantive systems both have a Customer #2 record). *iss* is determined by the source system identifier selected using the General tab of the Data Store dialog box (Figure 49, on page 178).

- The column *dimension_key_REAL* (where *dimension* is the dimension column name) is set to the first key value for each *sskey*. In other words, when a new *sskey* is discovered, then the *REAL* key is set to the new dimension key value. Subsequent dimension rows for this *sskey* will retain the original *REAL* key value.
- The UNKNOWN *sskey* always maps to dimension key value 1 in the Mapping table.

Note the following:

- Do not allow dimension column values to *oscillate* unpredictably. (See “First Dimension Value” on page 314 for more information.) In particular, do not rely on *key* filtering of duplicate *sskey* values if two or more rows during a single extraction might have different values for one or more dimension columns. The reason is that a new row will be created in the dimension table for every extraction for which a change is recorded. This can cause two values to “compete” with each other, forcing an unending sequence of row creation in the dimension table.
- Rows can be removed from dimension tables after they have been extracted with an explicit delete or truncation, or through use of the Initial Load Dimension.

Latest Dimension Value

The Latest Dimension Value updates rows instead of performing Slowly Changing Dimensions. It applies changes retroactively to a dimension. Thus the changes take effect for all historical data, as well as for current and future loads.

For example, assume that a sporting goods store has a category historically called *Rollerblades*. Now that they are selling other brands, the store wants to change the category to *In-line Skates*. By using the Latest Dimension Value semantic type, this change can affect all of the historical data because all previous sales of *Rollerblades* are now labeled *In-line Skates*. As a result, the store can compare year-to-year sales of all in-line skates.

First Dimension Value

This semantic type has the same duplicate *sskey* filtering as Slowly Changing Dimensions. Use this semantic type for an implementation that “restates history” when a source dimension table changes.

Note the following:

- New *sskeys* are inserted in both the dimension table and mapping table. Existing *sskeys* with one or more changed dimension columns are updated in place in the dimension table (that is, the same dimension row is used) with the latest values.
- In the EpiChannel log file, new *sskey* rows are reported in the *Inserted* column. The number of changed rows is reported in the *Modified* column. The number of rows in the staging table is reported in the *Processed* column.
- Latest Dimension Value has the same indexing as Slowly Changing Dimensions.
- The REAL column is always equal to the dimension key.
- Latest Dimension Value has the same UNKNOWN mapping behavior as Slowly Changing Dimensions.

First Dimension Value

The First Dimension Value semantic type ignores any changes to a dimension. The values that were present when the row was first inserted are preserved forever, regardless of any future changes. This semantic never modifies a row after it has been seen.

You might want to use this type if your source data comes from two systems that are not in complete agreement with each other. For instance, if one system has Customer #12345 as *David Anderson*, and the other has the same customer as *David Andersen*. Ideally, you would determine which one was in error and correct it.

In the meantime, you could choose to apply the first value read and to ignore the other. (This is a good method for avoiding the *oscillation* problem mentioned on page 313.) If you were to use the Slowly Changing Dimensions semantic type in this case, there would be a race between the two source systems for each extraction, and (in the worst case), your dimension values could alternate between the two values with every extraction.

Note the following:

- First Dimension Value has the same duplicate *sskey* filtering as Slowly Changing Dimensions.
- New *sskeys* are inserted in both the dimension table and the mapping table. Existing *sskeys* are ignored.
- In the EpiChannel log file, new *sskey* rows are reported in the *Inserted* column. The number of rows in the staging table is reported in the *Processed* column.
- First Dimension Value has the same indexing as Slowly Changing Dimensions.
- The REAL column is always equal to the dimension key.
- First Dimension Value has the same UNKNOWN mapping behavior as Slowly Changing Dimensions.

Initial Dimension Load

The Initial Load Dimension semantic type ignores the current Datamart entries. (It is also the fastest semantic type).

Initial Dimension Load loads dimension without regard to any previously existing rows. This can be used for the initial load of the empty EpiMart, and also to completely reload a dimension, ignoring existing values. Using any other semantic type would require emptying the existing dimension table before beginning the extraction.

Note the following:

- Initial Dimension Load has the same duplicate *sskey* filtering as Slowly Changing Dimensions.
- The existing dimension and mapping tables are ignored; all *sskeys* are imported directly into the dimension and the mapping tables.

Fact Semantic Types

- In the EpiChannel log file, new *sskey* rows are reported in the *Inserted* column. The number of rows in the staging table is reported in the *Processed* column.
- Initial Dimension Load has the same indexing as Slowly Changing Dimensions.
- The REAL column is always equal to the dimension key.
- Initial Dimension Load has the same UNKNOWN mapping behavior as Slowly Changing Dimensions.

FACT SEMANTIC TYPES

The fact semantic types are Count Unjoined (which replaced the fact semantic type Update Unjoined in Release 3.2), Custom Fact Index, Transactional, Transactional/State-like, Transactional/State-like/Force Close, Pipelined, Initial Load Fact, and Reload Max Date.

Count Unjoined (Optional)

Count Unjoined informs you of the number of rows that have been transformed by an outer join. Outer joins map fact staging table to dimension tables. This semantic type's usage of outer joins always prevents the loss of rows.

As of Release 3.3, Update Unjoined was renamed Count Unjoined because it no longer performs any updating. UpdateUnjoined searched for fact staging entries (in the dimension *sskey* columns) that did not map to any values in the corresponding dimension mapping table. It then set all unmapped values to the special token UNKNOWN. This was necessary because all of the other semantic types, such as Transactional and Pipelined, performed inner joins between the fact and dimension staging tables, which would cause a loss of data if this updating had not occurred earlier.

Releases 3.3 and later employ outer joins when mapping the fact staging table to the dimension tables. This usage of outer joins always prevents the loss of rows, whether or not Count Unjoined has been run previously.

Custom Fact Index

In EpiCenter Manager, you use the Custom Index tab of the Fact Table dialog box to define custom indexes for a fact table. The semantic type Custom Fact Index, however, must be executed for the fact table as part of the scheduled extraction process in order for these indexes to be built.

Important: Schedule the Custom Fact Index semantic instance *after* all other semantic instances for a fact table.

For a discussion of custom indexing, see “Custom Fact Indexing” on page 62.

Transactional

The Transactional semantic type is the simplest means of moving fact staging data into fact base tables. This semantic type uses the following logic:

- Only rows with *process_key* set to 1 (Transactional) are used; others are discarded.
- For *sskeys* that are already entered in the current fact base table, all rows in the staging table with that *sskey* and with the same or earlier *date_key* are discarded. If the *sskey* already exists in the fact table, only later dates can be added to the fact table.

Two or more rows with the same *sskey* can be imported into the fact base table if they arrive in the same extraction and the *sskey* either does not already exist, or exists but is dated earlier. (This property is used by the *pseudo-order* approach to the Booking/Shipping problem; see “Transactional/State-like/Force Close” on page 319.)

- In the EpiChannel log file, all rows added to the fact base table are reported in the *Inserted* column, and the total number of rows in the staging table is reported in the *Processed* column.

Note the following:

- Transactional semantics should be used for event facts; once the fact event has occurred, it can never be modified.

Transactional/State-like

- To reload transactions after they have been loaded into the fact base table, the rows must be deleted from the fact base table, or the fact base table must be truncated. See “Reload Max Date” and “Initial Load Fact” for more information.

Transactional/State-like

The Transactional/State-like semantic type allows changes to already existing rows in EpiMart. It uses the same logic as the Transactional semantic type, with the addition of the logic described below.

First these three steps occur:

- Step 1:** Records in the staging table with *process_key* of 2 (state-like) are treated as Orders that can be *differenced* between extractions (a discussion of this follows).
- Step 2:** For records in which *process_key* = 2, duplicate *sskeys* with the same *date_key* in the same extraction cause only the highest *ikey* value to be used. Other rows are discarded. This is the same filtering that happens in dimension semantics such as Slowly Changing Dimensions.
- Step 3:** For an *sskey* with the highest *ikey* (which means that for every set of rows with the same *sskey*, take the row with the highest *ikey*), if the *date_key* for that staging row is less than the last *date_key* for that *sskey* in the fact base table, the staging row is discarded. In other words, an Order can only be modified on its last reported date, or some time further into the future.

After Steps 1-3, the staging fact columns and dimension values are compared with the current values in the fact table (if any). Adjusting records are created in the fact table so that the fact table now reflects the reported *state* from the staging table. (This is why the term *state-like* is used.) *Differenced* transactions are invented if the numeric fact columns have changed.

If the dimensionality has changed, then the Order will be “de-booked” and “re-booked” with the correct dimensionality.

If the same *sskey* appears in the staging table with more than one *date_key*, then further adjusting transactions are made in the fact base table as appropriate to bring the fact base table in line with the reported staging rows.

Note: By convention, Bookings are entered with positive facts (negative for Returned Orders), whereas Shipments are entered with negative facts (positive for Returned shipments).

When using this semantic type it is difficult to ensure that Backlog calculations will remain consistent when Orders are not completely closed in the source system. Use the Transactional/State-like/Force Close instead.

Transactional/State-like/Force Close

This semantic type is equivalent to Transactional/State-like with the following additional logic.

Once a Booked Order is entered into EpiMart, it remains in that state (with an Open Backlog) forever. In normal scenarios, an invoice eventually arrives, which will close the Backlog. However, in some source systems, Booked Orders can be removed from the system completely. If such an Order had been entered previously into EpiMart, then it will remain in an Open Backlog condition forever.

The solution to this problem is to use Transactional/State-like/Force Close, which establishes a “Challenge Protocol” for Open Orders. In this scenario, all Open Bookings must be extracted into the staging table during every extraction. The reason is that the Force Close logic will close out all Open Orders (*sskeys* with non-zero facts in the system) that do not appear in the fact staging table. Only Booking Transaction types (those with *transtype_key* values between 1 and 99) will be affected in this manner.

When using this semantic type, the methodology that has been found to work in practice involves the use of pseudo-orders as follows:

- One extraction SQL statement extracts all Open Orders. Fact amounts are the Open amounts (what has not been shipped yet), not the ordered amounts. Transaction types for this statement should be in the Booking range (1...99), and *process_key* should be 2 (state-like).

Pipelined

- The second extraction statement uses *process_key* of 1 (Transactional) and represents all shipments in the system. These records normally go into the system with negative facts for positive shipments (by convention). These transaction types should be in the shipment range (101 or greater).
- The third statement is a restatement of the shipment, but as a Booking (*transtype_key* between 1...99). The *process_key* is still 1 (allowing the Transactional Semantics to import it), but the transaction type is a Booking. The same *sskey* and dimensionality as the second statement are used. These are the pseudo-orders since they are actual shipments that are entered as Orders.

The net effect of this methodology is that as a shipment is reported against Bookings, the Open Booking quantity in statement 1 is decremented, while the actual shipment is restated as a Booking transaction. Eventually, when the Order is removed from the system, the Force Close logic will close out any remaining Open fact quantities from statement 1 above. What remains in the system will be Shipments and their corresponding pseudo-orders. By construction, the Backlog will be zero.

Pipelined

Use the Pipelined semantic type for facts that can exist in several different life-cycle phases, called pipeline states; for example, sales opportunities or support calls facts.

The *transtype_key* field represents the pipeline. Before using this semantic type, you must define the pipeline stages and numbers.

Pipelined generates these transactions:

- Step 1:** When an *sskey* first enters a particular pipeline stage (*transtype_key*), then a positive fact row is created with that *transtype_key*.
- Step 2:** If that *sskey* subsequently moves backwards in the pipeline, then in addition to the new row created by the previous step for the new pipeline stage, a negative “de-booking” transaction is created with the old *transtype_key* minus 1. (This is a loss transaction.)

Step 3: If that *sskey* subsequently moves forwards in the pipeline, then in addition to the new row created in Step 1 for the new pipeline stage, a negative “de-booking” transaction is created with the old *transtype_key* plus 1. (This is a shipment transaction.)

Note: As a consequence of the +1 and -1 rows, the pipeline should be designed so that the *transtype_keys* that represents bookings in various stages are at separated by at least 3 integers.

For example, assume that the pipeline consists of the following steps: Prospect, Lead, Quote, and Order. You could set up *transtype_key* fields for this pipeline as follows: 302 = Prospect, 305 = Lead, 308 = Quote, and 311 = Order. The semantic type creates its own transactions for 301, 303, 304, 306, 307, 309, 310, and 311, which correspond to the forward and backward movements from the various pipeline stages.

To define measures that extract information such as Number of New Leads, you must define transaction types with these new names and keys (using EpiCenter Manager’s Configuration dialog box; see “Transaction Types” on page 288.)

Note: The pipelined semantic types ignores the *process_key* field because it is no longer relevant to it.

Initial Load Fact

Similar to the Initial Dimension Load, the Initial Load Fact semantic type is the fastest way to load a fact table. It also has the special property that the current table (whether A or B) is ignored so that this semantic type can reload an already-populated fact table. All existing rows, however, are lost. For this reason, Initial Load Fact is usually used during development when an installation is verifying whether a first extraction yields correct data.

Reload Max Date

Important: For first time extractions, you can use this semantic template in place of all other fact semantic types that load data when each *sskey* is being loaded into the staging table *only once*. This is because upon first extract, when an *sskey* is loaded only once, the special logic of the other semantic types (such as delta inference during Transactional/State-like) does not apply. If the first extraction does require multiple records per *sskey*, such as loading inventory values using Transactional/Statelike, then Initial Fact Load cannot be used.

Reload Max Date

The Reload Max Date fact semantic type is a hybrid of the Transactional and Initial Fact Load semantic types. Use it to reload a subset of an existing table in which the reload point is separable by date only.

Reload Max Date first determines the minimum date that occurs in the fact staging table. It then copies from the current fact table to the new fact table all existing EpiMart data that occurs with a date key prior to that minimum date.

The fact staging table is then used to hold all subsequent dates. Thus a complete load of data must be placed in the fact staging data from the minimum date forward. When loading the fact staging table (via SQL statements), you should use a WHERE clause based on a particular date.

APPENDIX G

EXPORT/IMPORT OF METADATA

All of the control information for an EpiCenter is stored in a single metadata repository called EpiMeta. EpiMeta represents a transactional, fully relational model of over one hundred and fifty tables, with complicated declarative referential integrity constraints. Epiphany provides EpiCenter Manager for configuring this metadata without the need to write to, or even know about, the underlying data structures.

Epiphany also provides a metadata Export/Import utility for moving metadata between EpiCenters and for backing up the definition of an EpiCenter. To use this tool properly, you should understand the basic metadata concepts discussed in this appendix.

METADATA OVERVIEW

All of the user-configurable metadata tables have integer primary keys. These non-natural keys are provided by the database engine when a metadata row is inserted; the value of the primary key itself has no intrinsic meaning. All relationships to this inserted row are made via this integer. For instance, the relationship between a filter group and its filter block is represented in EpiMeta via an integer column called *filter_block_key*.

Metadata Overview

The Access Export database does not simply contain a copy of the metadata tables being exported for this reason: If data were copied to the Export file, then when this same information is imported into a new EpiMeta, the integer primary key values in the Export file might clash with already existing primary keys in the target EpiMeta. Therefore, the Access database uses an EpiMeta-independent representation of metadata. See “Export File Format” on page 326 for a description of these Access database tables.

EpiMeta contains many tables, all of which are inter-related. Ticksheet metadata refers to constellation metadata (for instance, the attributes refer to dimension columns), while security metadata refers to ticksheets. In order to export only a portion of the metadata at a time, the Export machinery must decide where to stop exporting—otherwise, the entire metadata must be exported with each operation.

When using the Export metadata command of EpiCenter Manager, you must select which part of the metadata to export. Figure 83 shows an overview of the various domains of metadata within EpiMeta.

Each rectangle in the figure represents one of the options available for export. Everything within the rectangle is actually contained as metadata in the Export file when that option is selected. The arrows in the diagram represent references to other metadata. These references are contained in the Export file as well, but the references are made to other objects by name only. In other words, when exporting ticksheets, the measure mapping metadata is exported in the Export file, but the measures themselves are not exported (unless you also select the Measure option). Only a reference to the appropriate measures is exported.

Upon import, these references are used as follows. When ticksheet metadata is imported into a different EpiMeta, the import machinery searches for measures with the referenced names. If it finds these, then the same relationships are established in the new EpiMeta as the ones that were exported. However, if the Import machinery does not find these measures by name, the measure mapping information will be lost upon import.

Replacing Existing Metadata on Import

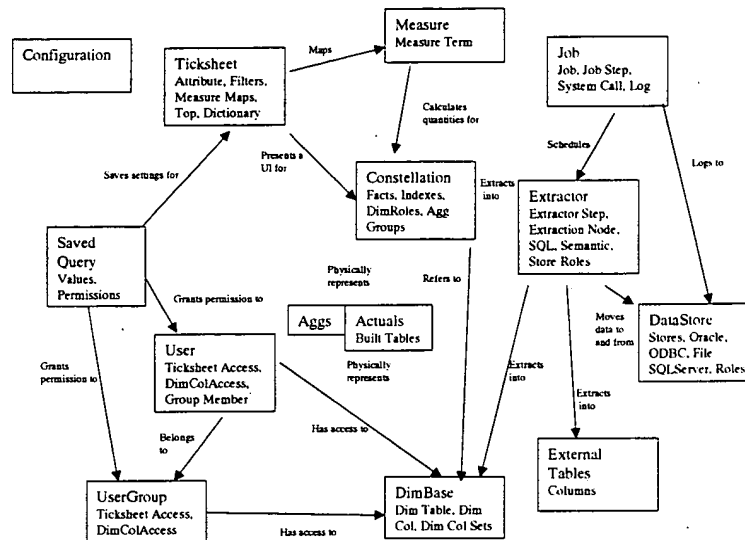


Figure 83: EpiCenter Data Model

Replacing Existing Metadata on Import

When importing metadata into an existing EpiMeta, the Import machinery will detect an attempt to overwrite existing metadata. Existing top-level objects are overwritten in place. Their children are deleted, however, before being recreated. The definition of “existing” is usually based on a unique name column for one of the metadata tables. For instance, tick sheets must have unique names, so an attempt to import a tick sheet with the same name as an existing one results in a warning message (unless the Always Replace option has been selected previously).

Actuals and Agg Metadata

Actuals and Agg Metadata

The Export All command in EpiCenter Manager does not export the entire contents of EpiMeta. The options for Actuals and Aggs are omitted by default from this export. This is because these sections of metadata are “derived” metadata: the Adaptive Schema Generator produces Actuals metadata, and Aggbuilder produces Agg metadata. As a result, a new EpiCenter can be constructed using the Export All metadata by rebuilding the schema, re-extracting, and re-running Aggbuilder.

To “clone” an EpiMeta in such a way that EpiMart itself will not need any modification when it is used with this new EpiMeta, you must select the options for Actuals and Aggs.

Note: An Export All operation on a running system, followed by a re-import of that metadata causes all Actual and Agg metadata to be lost.

Export File Format

Each Microsoft Access Export database has the same schema. In fact, this schema can be thought of as a meta-schema for representing relational data. For a description of these tables, see Table 13, “Export Tables,” on page 327.

To modify the row contents contained in an Export file, edit the *Export_col* table, which is simply a collection of name/value pairs for columns.

Table 13: Export Tables

Table Name	Description
Export_tbl	One row per metadata table being exported.
Export_row	One row per metadata row being exported.
Export_col	One row per column per row of metadata being exported. Only non-relationship columns are contained in this table.
Export_rel	One row per relationship between two rows of metadata. Can be a relationship between two rows contained in the Export file, or between one row in the Export file and one reference to a row in a foreign EpiMeta.
Export_status	Header information about the Export file.
Rel_parent	A reference to a metadata row in the foreign EpiMeta.

Export File Format

APPENDIX H

TROUBLESHOOTING

This chapter describes the Epiphany error conditions and error messages and suggests action you can take to resolve problems.

If you need additional assistance, please e-mail Epiphany Customer Support: support@epiphany.com.

REGISTRY EDITOR WARNING

Using Registry Editor incorrectly can cause serious problems that may require you to reinstall your operating system. Microsoft cannot guarantee that problems resulting from the incorrect use of Registry Editor can be solved. Use Registry Editor at your own risk.

For information about how to edit the Registry, view the “Changing Keys And Values” Help topic in Registry Editor (**Regedit.exe**), or the “Add and Delete Information in the Registry” and “Edit Registry Data” Help topics in **Regedt32.exe**.

Note that you should back up the Registry before you edit it. If you are running Windows NT, you should also update your Emergency Repair Disk (ERD).

SQL SERVER ERROR MESSAGE

Cannot Connect to the Server

The most likely cause of this problem is that the ODBC driver for SQL Server has not been correctly installed.

You can verify this by opening Start\Settings\Control Panels\ODBC\ and selecting the tab for the ODBC Drivers. Make sure there is an entry for SQL Server. If there is not, you can use the appropriate CD-ROM to reinstall it:

SQL Server 7.0: The ODBC driver is installed with the SQL utilities on the SQL Server CD-ROM.

APPLICATION SERVER ERROR MESSAGES

Note: If the Application Server won't start, you can run the Scrutiny debugging tool in EpiCenter Manager to diagnose the problem. See "Running the Scrutiny Debugging Tool" on page 223.

The following Application Server errors messages are described:

- "Cannot Connect to the Server" on page 330
- "User Cannot Log In" on page 331
- "Service Control Manager Fails to Start Application Service" on page 333
- "Out of Memory" on page 334
- "Invalid Object DATE_O Error" on page 334
- "Not a Valid Application Error" on page 335
- "Internal Windows NT Error" on page 335
- "EpiQuery Engine Database Connection Open Failure Exception" on page 336
- "Charts Do Not Display" on page 337
- "GIF Images Fail to Display on Web Pages" on page 337
- "No Results Available for a Query" on page 338

- “Result Page Error: Extraction Date Unknown” on page 339
- “Web Server Message: Object Not Found” on page 340
- “Browser Crashes When Retrieving Results from Application Server” on page 341
- “Refresh Program Fails” on page 342
- “Application Log Full Error” on page 342
- “Application Server Log Security Problem” on page 343

User Cannot Log In

There are three types of failure associated with an unsuccessful login to the Application Server:

Step 1: The Application Server generates a critical exception indicating an application error, or an error related to the NT security domain).

If this case, an error message appears in the browser with a link to the log that contains the stack trace for the error. Read the error description. Generally, it relates to the way NT domain security is configured or set up at this point. For example, the error might say that the domain controller could not be reached.

Try to fix the problem based on the specifics of the error message. If you need additional assistance, please e-mail Epiphany Customer Support: support@epiphany.com.

Step 2: An invalid login message displays even though there is a valid username and password.

This error may result when the search for the username/password occurs on the wrong machine. For example, suppose a user *foo* existed on both the machine local to the Application Server and the primary domain controller.

The order in which the user *foo* is searched is as follows:

- the local SAM
- the primary domain

User Cannot Log In

- the trusted domains

Thus, user *foo* in the local SAM will be found first, even though the username/password combination could have been for the user *foo* that exists in the primary domain or in a trusted domain.

To solve this problem, further specify the username by including the domain name before the username; for example, *EPIPHANY\foo*. In general, specify the full user's name if the user's browser displays an invalid login message.

If the error persists, please e-mail Epiphany Customer Support: support@epiphany.com.

Step 3: Authentication was successful, but the user is not allowed to use the Epiphany system.

For a user to have access to Epiphany System, he or she must be a member of at least one Epiphany group after a sync-up process.

Use EpiCenter Manager to check group memberships after you receive this error message. If the old group memberships were removed, then this error has occurred because in the NT domain, the user is not a member of the NT groups that have the corresponding groups marked Synchronize in EpiCenter Manager.

Also check that the username used to log in matches the username in EpiCenter Manager. If the username in EpiCenter Manager is prefixed with domain name other than the one that the actual NT user is a member of, then the login could fail and return an error message to this effect. Specifying the full username upon log in may fix this problem.

If the error persists, please e-mail Epiphany Customer Support: support@epiphany.com.

Additional Action to Take If User Still Cannot Log In

Note: If you need to have a working system immediately, you may temporarily disable security by hooking up EpiPassThruLogon module. Be sure to read “**Application Server Security**” on page 250 for instructions on configuring authentication modules.

If you cannot fix the problem using the above procedures, e-mail a security log (**xxx-xxx-xxx-SECURITY.txt**) and Application Server log (**xxx-xxx-xxx-SRV**) to Epiphany Customer Support.

A description of the three security logs follows:

APPSERVER.LOG The only file with a suffix of **.log**, this is the main Application Server activity log, which logs a summary of all queries performed on that server. Includes the user who submitted the query request, the type of ticksheet submitted, the failure or success of the query, and the amount of time for the query.

SAVE_RESTORE Logs all save and restore operations during a session.

QM_randomly_generated_alphabetic_sequence_username
Consists of individual logs, one per user, that show the content of each query or other browser transactions. These logs contain the selected attributes, measurement selections, filters, and so forth submitted with each query. It also contains the SQL sequences, the duration times of the various stages of the query, and the total time for each query.

Service Control Manager Fails to Start Application Service

If after installing an Application Server, the Service Control Manager (Start Menu\Settings\ControlPanel\Services) fails to start the Application Server, then double-click the service in question, and make sure that Allow Service to Interact with Desktop is selected.

Out of Memory

Out of Memory

The Application Server requires sufficient memory (at least 64 megabytes) to function properly. The default `jre` program allows the Application Server to allocate only 16 megabytes of extra memory. Queries that involve Product and Customer (or any large dimension) and have the Rows option set to All can easily exhaust the default 16 megabytes. When this happens, you will receive an Out of Memory exception error: `java.lang.OutOfMemoryError`.

The solution to this problem is to make sure that you have started the Application Server with the `-mx64M` parameter. For example:

```
jre -mx64M -classpath ... com.epiphany.server.Server ...
```

The `-mx64M` parameter allows the Application Server to allocate up to 64 megabytes of memory.

The Refresh program can also exhaust memory if run three times consecutively. This is because the atomic switch from old to new metadata requires a least two copies of the metadata to be alive at the same time. Until all references to the old metadata are released, both objects stay in memory. Eventually, as sessions are cleaned up and as old command threads die, the old object will be released.

VirtualMartDatabase Key Missing Error

If you receive this message from the Application Server, more than likely a valid EpiCenter data store has not been specified.

When you configure an EpiMeta database using EpiCenter Manager, you will use the EpiMart Data Store dialog box in the Extraction\Data Stores folder to specify the target EpiMart. Click the Properties tab and make sure that the data store name is correct.

Invalid Object DATE_O Error

When starting the Application Server, you may encounter the SQL error **Invalid object Date_0**. This error is generated when the Application Server attempts to read the metadata (EpiMeta).

Not a Valid Application Error

To correct this problem:

- Make sure that the EpiMart data store specified via EpiCenter Manager points to a valid EpiMart database. (Open the Extraction\Data Stores\EpiMart Data Store dialog box, and click the Properties tab.)
- Make sure that you have populated the date dimension in your EpiCenter using EpiCenter Manager. To verify this, go to ISQL and use the **sp_help** command on your EpiMart database. Check for the existence of the *Date_0* table. If it is not there, it has not been populated. See “Getting Started” on page 87 for instructions.

Not a Valid Application Error

This problem can occur for two reasons:

- If a service component required for Windows NT, an application, or a network protocol, is corrupted or missing.

To correct this problem, manually expand the service component file. For example, if the service name in Event ID 7000 is MUP, expand MUP.SY_ from the Window NT CD-ROM to MUP.SYS in the %SystemRoot%\SYSTEM32\DRIVERS folder.

- If the folder location of the executable contains spaces in the directory name (that is, has a long filename). An example would be when the executable is located in the \Program files\service.exe folder.

To correct this problem, modify the Registry key that contains the executable path so that it is enclosed in quotation marks.

Internal Windows NT Error

This error results when the Epiphany Application Server cannot be started as a service. The exact error message that was returned from the Epiphany Application Server is logged in the Windows NT Event Log.

EpiQuery Engine Database Connection Open Failure Exception

To locate this error message in the Event Log:

Step 1: Go to Start menu\Programs\Administrative Tools (Common)\Event Viewer.

Step 2: Click the Log menu and select the Application.

Step 3: Double-click the appropriate event.

All Epiphany Application Server events have the source EpiAppServer.

To solve this problem, first stop any running Application Server services. Then log onto Windows NT as an administrator and re-install the Epiphany software. If this does not solve the problem, start the Application Server from the command line as described in “Running as a Console Application” on page 230. The console output should describe what is wrong.

EpiQuery Engine Database Connection Open Failure Exception

An EpiQueryEngineDBConnOpenFailureException from the EpiQueryEngine means that the Application Server is having difficulty connecting to the database. Take these steps to correct this problem:

- Make sure that the username and password for the EpiMeta database are set correctly in the Windows Registry.
- Make sure that the database name and server are also configured properly (again, in the Registry).
- Make sure that SQL Server TCP/IP Sockets have been enabled. Usually, this option is set when the SQL Server is installed on the machine. To verify that these sockets have been enabled, follow these steps:

Charts Do Not Display

- **SQL Server 7.0:** From the Start menu, open Microsoft SQL Server 7.0 and select Client Network Utility. Select the Network Libraries tab in the SQL Server Client Network Utility dialog box. TCP/IP needs to be configured as client for Epiphany to connect to the SQL Server. To configure TCP/IP:

Step 1: Select the General tab in the SQL Server Client Network Utility dialog box.

Step 2: Click Add.

Step 3: Select TCP/IP in the Add Network Library Configuration dialog box.

Step 4: In the Server alias box, enter the alias of the computer that runs SQL Server and listens on the Windows Sockets Net-Library.

You can also specify the server with its IP address instead of its name. See *Microsoft SQL Server 7.0 Books Online* for more information.

Charts Do Not Display

If there is a broken link to an image, or no image appears, or an image with an error message appears instead of the chart, there is a charting problem. Please contact Epiphany Customer Support.

GIF Images Fail to Display on Web Pages

If GIF images do not appear on your Web pages, do the following:

Step 1: Make sure that your Web server has an alias for your *instance_name* that points to a valid directory. If you performed a normal installation, then all Web files should be located in the directory:

C:\Program Files\Epiphany\Instance_name\web\WWWROOT

and GIF files should be located in the directory:

C:\Program Files\Epiphany\instance_name\web\WWWROOT\images

(assuming your Epiphany Application Server was installed in

C:\Program Files).

No Results Available for a Query

If your Web server is IIS 3.0, you can find the aliases by opening up the IIS Internet Service Manager (normally this is located in your Start\Microsoft Internet Information Server menu) and selecting the Directory tab. Make sure that there is an entry for *instance_name* that has a valid directory.

For the IIS 4.0 Web server, open the Windows NT Option Pack\Microsoft Internet Server\Internet Service Manager. Aliases are listed in the IIS Management Console.

- Step 2:** Make sure the **WWWROOT\images** directory has the GIFs in it. If you are missing a few GIFs, then you will need to reinstall your Application Server or copy the GIFs from a different instance.
- Step 3:** Check the BASE HREF tag that is defined in the source of the page. In your browser, try to view the source for this HTML page. Look for the BASE HREF tag at the top of the page. Note what it is and make sure that it is a valid alias using the procedure above.
- Step 4:** Make sure your Web server is serving pages and that your browser is not displaying cached HTML. Clear the caches (Memory and Disk) on your browser, close your browser, and try to access the URL again. Also, try referencing another URL from your Web server to make sure that it is running.

Technical Note: All of the GIFs on an Application Server-generated page are referenced from the **images** directory, which is relative to the BASE HREF specified at the top of the page in a META tag. The *instance_name* is derived from the URL that you use to access the Application Server. It is used throughout the system to read the correct Registry entries and to generate the correct URLs.

No Results Available for a Query

The most common cause of this problem is that the *current_datamart* value in the EpiMeta database is set to the wrong database.

Result Page Error: Extraction Date Unknown

Use EpiCenter Manager to make sure that the *current_datamart* is set to the correct database. You can also use the ISQL tool to determine which database, A or B, has the most current data.

Result Page Error: Extraction Date Unknown

The *last_extract_date* is a field that is kept in the EpiMeta database. It is used to keep track of the date displayed on the top of all Clarity and Relevance reports as the date of the last extraction. It is normally populated by the extraction SQL entered in the End of Extraction job in the EpiCenter Manager. It can also be populated by EpiCenter Manager via the Configuration dialog box. This field must be entered in one of two very strict formats. The default format is *mm/dd/yyyy*; for example: 01/14/1998.

The Application Server applies the following logic to parse the date:

Step 1: If the field has more than 10 characters, then parse it using the pattern *mm/dd/yyyy hh:mm:ss*. Otherwise, use the pattern *mm/dd/yyyy*.

Step 2: If the parse fails, then use {extraction date unknown}.

In addition, the date that is displayed at the top of the report always has a time zone. The time zone is printed based on the default time zone of the machine on which the Application Server is running. The date that is being displayed is also taken into consideration. For example, the date 12/20/1997 will display as December 20, 1997 PST if the Application Server was running on a machine in California. However, the day 05/12/1998 will display as May 12, 1998 PDT on the same machine since Daylight Savings time took affect in April.

Note: EpiCenter Manager does not allow the user to enter a date in the format *mm/dd/yyyy hh:mm:ss*. Only the SQL in the extraction job can enter dates of this format, or you can manually arrange this via ISQL.

Web Server Message: Object Not Found

Web Server Message: Object Not Found

If you installed the Epiphany software using the standard Epiphany software installation program, you will access your Web server through this type of URL: **http://machinename/scripts/instance_name/Epiphany.dll**.

If you receive an object not found error message, follow these steps:

Step 1: Start and stop the Web server. If this does not solve the problem, go to the next step.

Step 2: Verify the Web server is serving pages.

Note: Make sure that your browser is not just returning cached HTML pages by clearing your memory and disk cache before testing.

Try to access other URLs from the same *machinename*. Try to access other static HTML files that are installed as a part of the Application Server installation, such as **http://machinename/instance_name/clarityhelp.html**.

Step 3: If this does not work, try accessing any other file that the Web server should be serving. Consult the Internet Service Manager for the names of other aliases that the Web server should be serving, and then try to access these aliases with your browser.

In most cases, your Web server searches the **C:\Inet-pub\scripts\instance_name** directory to find the **Epiphany.dll** program. Make sure that there is such a directory on your machine, and that the **Epiphany.dll** file is in that directory.

Step 4: Check the file permissions for the **Epiphany.dll** file.

First, make sure that the account IIS uses for anonymous logins has file access permissions for the **Epiphany.dll** file. Go to the IIS 3.0 Internet Service Manager and look at the Anonymous Login account box. In IIS 4.0, right-click the name of the machine and choose Properties. Select the Direc-

Browser Crashes When Retrieving Results from Application Server

tory Security tab. In the Authentication Methods dialog box, select Allow Anonymous Access and click the Edit button to modify the account used for this purpose. Make sure the user and password are correct.

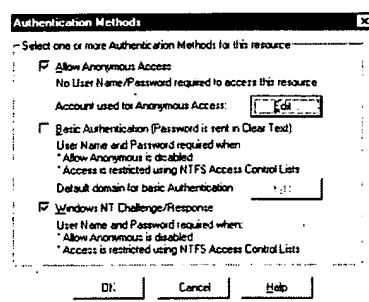


Figure 84: Authentication Methods Dialog Box

To check file permissions, open the Windows NT Explorer window and right-click the **Epiphany.dll** file in the `./scripts/instance_name` directory specified above. Go to Properties and open the Security tab. Click View Permissions and make sure that the account that you used for Anonymous Web Login has access to this file. (If Everyone is selected, then all people, including the Web login account, have access to the file.)

Browser Crashes When Retrieving Results from Application Server

In general, for machines with less than 32 megabytes of RAM, the browser will perform very poorly when parsing HTML files that are larger than 150 kilobytes. Therefore, users who do not have at least 32 megabytes of RAM installed on their machines should refrain from retrieving large queries.

Refresh Program Fails

An example of a large query follows. Suppose you query Customer by Fiscal Year and apply the filter Business Unit: Learning. This query returns approximately 3800 customers, and the HTML that is generated is 1.8 megabytes. When loaded in Netscape 4.03, it occupies 37 megabytes, takes 3 minutes to parse, and consumes the entire processor. The parsing is the bottleneck in the downloading of the file.

Note: When started, Netscape Navigator 4.03 requires 8 megabytes immediately to load whatever it is loading. Microsoft Internet Explorer (IE) requires 6.7 megabytes, and is substantially faster at parsing the text.

Refresh Program Fails

If the Refresh program fails, the Application Server will continue to use the old metadata information. Users will still be able to log in and view the old ticksheets. This is because the Refresh program reads the new metadata into temporary structures that are atomically exchanged with the old structures only if all of the refresh structures were read correctly.

To solve this problem, rerun the Refresh program.

Application Log Full Error

If you receive this error, you can take the following action to delete all log events.

Note: If you do not want to delete all of the log events, you can also increase the amount of space allocated to the application log. You can also select the Override as Needed option in the Log Properties dialog box.

Step 1: Open Start\Programs\Administrative Tools (Common)\Event Viewer.
Select the Application Log under the Log menu item.

Step 2: Select Clear All Events from the Log menu.

When prompted whether you want to save the log files, and whether you really want to delete all of the log events, respond in the affirmative.

Application Server Log Security Problem

The **APPSERVER** and **SRV** logs contain all of the session IDs that have logged in. Although unlikely, it is possible that someone could alter a session ID (along with a fake IP address) and run queries against the Application Server.

The log file also contains information about who is running a query, and when they are running a query, which may be confidential information. The log files for each query contain all the information necessary to rerun a query. Hence, a malicious party can also determine the type of queries run by a particular user.

It is important to give user access to the log file directory so that when problems occur, the Download query log link at the bottom of each results page will work. Directory browsing for the log file directory, however, should be turned off.

To turn off directory browsing, de-select the Directory Browsing Allowed option in the Directories tab of the WWW Services Properties dialog box in the Internet Service Manager. (Normally, IIS is located in your *Start\Microsoft Internet Information Server* menu.)

Application Server Log Security Problem

GLOSSARY

Actual Tables

The Actual tables are metadata created by the Adaptive Schema Generator after it has built or modified tables in EpiMart. The Actual tables are consulted on subsequent schema generations to determine delta operations to perform. Also, other tools examine these tables as a check to ensure that EpiMart's schema matches the current metadata definition.

Adaptive Schema Generator

The Adaptive Schema Generator is a component of EpiCenter Manager that builds the tables in EpiMart using the schema metadata definitions in EpiMeta. When schema metadata changes, this program can perform delta operations to modify the schema without losing data in EpiMart.

Aggregate Builder (Aggbuilder)

Aggbuilder is the executable program (**agg.exe**) that builds aggregate tables in EpiMart. Normally, the execution of this program is scheduled after all data has been extracted from source systems and merged into EpiMart.

Aggregate Group

An aggregate group is a metadata definition of the fact aggregates to be built. It is a series of instructions that tell Aggbuilder which aggregates to build on one or more fact tables. An aggregate group applies to a single constellation. The actual fact aggregates that get built are determined by a combinatorial expansion of the instructions in the aggregate group.

GLOSSARY

Aggregate Navigation

Aggregate navigation is the process by which the Epiphany query machinery determines the optimal aggregate table to use to satisfy an application's request for information. Applications such as Clarity do not need to know about the presence of aggregates because the aggregate navigation process makes the aggregate layer abstract.

Application Server

The Epiphany Application Server is the Windows NT Service that connects with a Web server and serves up Epiphany ticksheets and reports. An Application Server is configured to connect with an EpiCenter.

Attribute

An attribute is a dimension selection on a ticksheet. The row and column lists in Clarity contain attributes. Attributes are related to exactly one dimension column in a table. The only difference between an attribute in a table and one in the Epiphany system is that an Epiphany attribute has an associated display label, such as Fiscal Year, that can be configured via EpiCenter Manager.

In Relevance ticksheets, an end user may select attributes in order to forecast trends, or to analyze highs and lows, best and worst cases, or graphs for various aspects of your business.

The Momentum attributes are derived from the group or individual demographic dimension tables.

Attribute Role

An attribute role is the usage of an attribute on a ticksheet. Attributes are defined only once per ticksheet. For example, if you want a single attribute to appear in both the rows and columns listboxes of Clarity, when configuring the ticksheet, assign that attribute both of these roles: Clarity Row and Clarity Column.

Backlog Type

Backlog type is used on a measure term to specify that the measure term should exhibit accumulation behavior. When time is one of the dimensions of a query, the results for different time periods will exhibit beginning or ending accumulated values instead of the actual transactions that occur in that time period. BEGIN specifies beginning accumulated value, whereas END specifies ending accumulated value.

Base Dimension

A base dimension is a physical dimension table in EpiMart, such as Customer and Product. Base dimension tables contain the actual dimensional values that are available for reporting or filtering. Base dimension tables can be used more than once in a single constellation via dimension roles. Base dimensions are defined for the entire EpiCenter and can be shared by all of the constellations within the EpiCenter.

Base Dimension Aggregate

A base dimension aggregate is a physical table in EpiMart that represents an aggregated view of a base dimension table. A base dimension aggregate results from the removal of one or more columns from the base table, followed by the removal of duplicate rows caused by this deletion.

Base Table

Base tables are the EpiMart tables (both fact and dimension) that store non-aggregated customer data at the level of granularity of extraction. These tables will be used as input to Aggbuilder to build aggregate tables with the same information at different levels of granularity. Base table names end with an underscore zero (_0); for example, Order_0_A or Order_0_B.

GLOSSARY

Constellation

A constellation is a grouping mechanism for like-structured fact tables. A constellation defines the dimensionality of all fact tables in that constellation. Dimension roles and degenerate dimensions are defined for a constellation, not a specific fact table. All fact tables placed in that constellation then inherit that definition. Both ticksheets and measures are defined within the scope of a single constellation, and thus apply only to the dimensionality defined by that constellation.

The Momentum constellation can be associated with multiple Momentum Adjacent Constellations. (See *Momentum Constellation* and *Momentum Adjacent Constellation*).

Custom Index

A custom index is the metadata definition of indexes to build on fact tables in EpiMart. Normally, each fact table is indexed by the leading term *Date*, which works only for queries that are selective by date. You may use EpiCenter Manager to specify additional indexes to build. To actually build these indexes, however, you must apply the semantic template Custom Fact Index.

Data Store

A data store is a logical location of data to be used either as a source or sink within an EpiCenter. Data stores include relational database connections, and files and directories.

Dataset

A dataset is a logical user-interface grouping of ticksheets that display the same view of data. Typically, different ticksheets types (such as Clarity and Relevance Profiling) that show the same data within a constellation are placed within a dataset. The dataset appears to end-users as a named entity that corresponds to a single business process. For instance, a dataset called *Executive Summary* might be defined that shows similar amounts of high-level data, using different Epiphany applications.

Date Dimension

The Date dimension is a special base dimension table supplied by Epiphany, which is used for storing all attributes related to time. All fact tables in an EpiCenter receive the foreign key *date_key* to this table.

Degenerate Dimension

A degenerate dimension is a single column dimension in a fact table that stores textual information in lieu of using a foreign key to a base dimension table. Degenerate dimensions are used when a single field of data fully specifies a dimension of that fact. Examples include Invoice Number and Serial Number.

Because degenerate dimensions appear only in fact base tables, they are never aggregated. (They are, however, always indexed.) Degenerate dimensions are defined on a constellation rather than for a specific fact table. All fact tables within that constellation inherit the degenerate dimension definition.

Dimension Column

A dimension column is a single column or attribute of a base dimension table. It contains the actual customer data that appears on reports or in filters.

Dimension Column Access

Dimension column access is the ability for a user or group to view only a subset of the available data in an EpiCenter. For example, dimension column access settings can be used to limit specific users to only data for the Western Region.

Dimension Column Set

A dimension column set is a named set of dimension columns (all within a single base dimension) that defines which columns will be used in a base dimension aggregate.

GLOSSARY

Dimension Role

Dimension roles allow a single base dimension table to be used in different roles within a constellation; for example, a Territory base dimension table includes Eastern Sales and Western Sales data. A Western Sales dimension role would reference the Territory base dimension for its data.

A dimension role is a dimension column in a fact table that defines the foreign key that references a base dimension table. It always has an associated base dimension table. Dimension roles are defined within a constellation, and all fact tables in that constellation inherit the dimension role. Multiple roles within a single constellation can refer to the same base dimension.

The Momentum constellation has two special dimension roles: *group* and *indiv*, which are described below.

Demographic Base Dimensions and group and indiv roles (Momentum)

The demographic base dimension tables for individuals and groups are used by Momentum to generate lists of individuals and groups. A Momentum primary constellation must have one demographic base dimension table pointed at by the *group* dimension role. It can also have a base dimension table pointed at by the *indiv* dimension role. (These dimension roles cannot point to the same base dimension table.)

The base dimension table associated with the *group* dimension role must contain group demographic data, and the base dimension table associated with the *indiv* dimension role must contain individual demographic data; for example, account holders in a household, or employees in a company.

EpiCenter

An EpiCenter is a single logical Epiphany database installation that includes customer data in addition to control metadata. It physically consists of two databases: EpiMeta and its associated EpiMart.

EpiCenter Enterprise Manager (EpiCenter Manager)

EpiCenter Enterprise Manager is a Microsoft Windows program (EpiMgr.exe) for configuring EpiCenters. These key components of the Epiphany system are available via EpiCenter Manager: schema, ticksheet configuration, extraction, security metadata, the Adaptive Schema Generator, and metadata export/import.

EpiChannel

EpiChannel is the Epiphany program (**extract**) that executes extraction jobs. The person who runs EpiChannel enters parameters to connect to an EpiCenter and specifies jobs to be executed. EpiChannel then implements the extraction steps declared as metadata in the EpiCenter.

EpiMart

The physical database that contains actual customer data. The schema of tables in EpiMart is determined by the metadata in EpiMeta. The data contents of EpiMart tables are determined by the extraction steps in EpiMeta.

EpiMeta

EpiMeta is the metadata repository for an EpiCenter. It contains all control information for the EpiCenter and therefore defines the behavior of that EpiCenter. EpiMeta points to EpiMart via the special EpiMart data store, which is available in every EpiMeta.

Epiphany Report URL

The URL that is embedded in a mail message and invokes an Epiphany query.

External Column

An external column is a single column in an external table.

GLOSSARY

External Table

An external table is a table built in EpiMart, usually as a temporary table used during extraction. External tables must be declared as metadata (as opposed to begin built outside of Epiphany) in order for the Adaptive Schema Generator to know of their existence. Otherwise, the table would be purged by the Purge EpiMart command.

Extraction Group

An extraction group is a set of extraction steps.

Extraction Step

An extraction step is a single atomic extraction operation. It can be either a SQL statement or a semantic Instance.

Extractor

An extractor is a list of extractor steps together with input and output data stores. It logically specifies a series of steps that move data from the input to the output data store.

Extractor Step

An extractor step is a single step of an extractor, which always points to an extraction group. In this way, extractors consist of an ordered list of extraction groups (which are ordered lists of extraction steps).

Fact Aggregate

A fact aggregate is a physical table in EpiMart that contains aggregated fact information for a single fact table.

Fact Clusters

Fact Clusters are tables MomentumBuilder builds to speed up runtime performance. When you associate a fact table in a Momentum Adjacent Constellation with a mini-dimension, the system creates copies of the fact table sorted by the mini-dimension.

Fact Column

A fact column is a single numeric column in a fact table.

Fact Counts

Fact counts are tables MomentumBuilder builds to speed up runtime performance. Associating fact tables in a Momentum Adjacent Constellation with a mini-dimension and a specific transaction type, such as Booked or Booked Returns causes statistics to be kept that help in selecting appropriate fact clusters.

Fact Table

A fact table is a physical table in EpiMart that contains numeric data in addition to references to dimension tables that specify attributes about the fact, such as who, what, when, and where the fact occurred.

Filter Block

On a ticksheet, a filter block controls the user's ability to filter on a single dimension column (for a dimension role of the ticksheet's constellation). The filter block also defines the appearance of the filter (for example, check box versus listbox).

Filter Element

A filter element is a single check box for filtering within a filter group, or a single entry within a listbox filter block.

Filter Group

A filter group is a logical grouping of filter elements within a filter block. It is applicable only to check box filter blocks.

Glossary Entry

A glossary entry provides end users with online definitions of the terminology used on a ticksheet, such as Units, Gross, and Sell-Through, which are hyperlinked to a glossary page. The creator of the ticksheet configures these entries via EpiCenter Manager or Web Builder.

GLOSSARY

Influence

Influence is one of the Relevance ticksheet types. Influence builds models of a company's data in order to identify the factors with the greatest impact on the business. For example, how do various attributes of a household influence whether or not a member purchases a product, and What influences whether a prospect responds to a direct mail offer?

Ind_Group_Joiner

This is a special fact table for Momentum that serves to connect the individual and group base dimension tables. It does not function in the same way as other fact tables in the Epiphany system. Although you cannot modify this dialog box, you do need to define how to extract data into the *Ind_Group_Joiner* table.

Job

A job is a top-level workflow object that defines a sequence of steps to be performed by EpiChannel. It also specifies the logging locations for that execution.

Job Step

A job step is a single step of a job, which can be either an extractor or a system call.

Leaf Dimensions

The dimension tables in a Momentum Adjacent constellation that point to non-demographic base dimensions.

Lists (Momentum)

Lists of demographic row keys that result from an end-user query, which reside permanently in EpiMart.

Measure

A measure is a single business calculation. It can be an arithmetic combination of measure terms using RPN (Reverse Polish Notation).

Measure Mapping

The mapping of ticksheet selection columns to measures.

Measure Sets

A measure set is used to define the target variable predicted by Influence. Measure sets are collections of other objects (measures and attributes). All measure sets include a Count measure, and most measure sets additionally include either a Sum measure or an attribute.

Measure Term

A measure term is a single component of a measure. It refers to the aggregation of a single fact column, such as *SUM(Order.net_price)*, with a particular transaction type. It can be combined with other measure terms to create a composite measure (business calculation).

Mini-Dimensions

A *mini-dimension* is a subset of a base dimension table used for Momentum queries.

Momentum

Momentum allows you to create lists of demographic dimensions, such as Customers, Resellers, Sites, Households, Individuals, Contacts or other similar entities. A list is always generated by using the Momentum user interface to apply successive filtering. Once you have defined a filter in Momentum, you can find out how many matches it corresponds to in the database, either exactly or approximately.

Momentum filters can be for individual data only, or for both individual and group, or household data. You can filter on attribute data and, optionally, on transactions. A transaction is usually an action such as bought, returned, called call center, received promotion or responded to campaign.

GLOSSARY

Momentum Adjacent Constellation

The Momentum Adjacent Constellations allow you to generate lists based on behavioral or transactional aspects of your customers. Momentum adjacent constellations must have at least one dimension role pointing at a Momentum demographic base dimension table. See *Demographic Base Dimensions and group and indiv roles*.

MomentumBuilder

MomentumBuilder is the executable that creates special Momentum tables, such as mini-dimension tables and clusters.

Momentum Constellation

The Momentum Constellation is the special Momentum constellation that has the Momentum fact table *Ind_Group_Joiner*, and the group dimension role. The Ticksheets folder for configuring Momentum ticksheets is part of this constellation. It allows you to generate lists based on demographic data about your customers.

Pull/Push SQL Statement

A pull/push SQL statement is a type of extraction step that issues SQL against an input data store and then pushes the result set into a table in the output data store of an extractor.

Queryable Fact Column

A fact column upon which a user can query transactions using Momentum.

Query Machinery

The component of the Epiphany Application Server that communicates with EpiMart and actually issues SQL statements against the DBMS. The query machinery is a common component of Epiphany's front-end applications.

Relevance

Relevance ticksheets enable users to analyze a company's data in order to find trends, project quarterly results, profile and segment customers, and find correlations and anomalies in the company's data. Also see *Influence*.

Report

A report is a saved set of ticksheet settings from a previous query that can be viewed again. Users and groups are granted permission to use or modify Reports. Users who request the same Report (and have the same dimension column access rights) will receive the same output (reports, charts, and so forth).

Report Gallery

The Report Gallery is a component of Epiphany's front-end user interface that allows users to view reports. EpiCenter Manager also contains a Report Galley in the Security folder that has additional features for administrative use.

Scrutiny

Scrutiny is a debugging tool available through EpiCenter Manager that you can run periodically, or whenever you suspect problems with the EpiMeta or EpiMart tables, or if the Application Server fails to start. Scrutiny loops through a series of tests to determine if any problems exist. If it finds one, it explains the problem, asks if you want to correct it, and then corrects it, if possible.

Security Manager

Security Manager (**SecMgr.exe**) is an Epiphany program for assigning access rights to log onto the system, as well as permission to view, modify, and save ticksheets. The Security Manager program, which is a subset of EpiCenter Manager, allows its users to access the security features of EpiCenter Manager, without having access to its full-functionality.

Selection Column

A selection column is a single column in the measure section of a ticksheet. The combination of one value from each selection column translates into a single measure in the report output.

GLOSSARY

Selection Column Element

A selection column element is a single value in a selection column. For instance, a selection column might contain the elements *Gross*, *Net*, and *Return*.

Semantic Instance

A semantic instance is a post-compilation SQL program. When a semantic template is applied to either a base dimension or fact table, the template becomes instantiated as an actual SQL program that can be used to accomplish specific business rules during extraction.

Semantic Template

A semantic template is a generic SQL program intended to accomplish specific business rules (these rules are referred to as semantic types) during extraction. A semantic template does not refer to actual column or table names. Only when the template is applied to a base dimension or fact table (via a semantic instance) does the SQL contained within it refer to real column and table names.

Semantic Type

A semantic type refers to the logical business process for which the semantic template is applied.

Source System

A source system is the logical notion of a source of business data. Two physical databases (one a backup of the other) might represent the same source system within Epiphany.

SQL Statement

A SQL statement is an extraction step that issues custom SQL against an input data store. The results of the SQL statement can either be discarded or used to push data into a table in the output data store.

Stand-Alone SQL Statement

A stand-alone SQL statement is a SQL statement whose results are discarded.

System Call

A system call is an extraction step that causes an operating system program to be invoked.

Ticksheet

A ticksheet is a form that end users open in a Web browser and use to submit queries to the data warehouse. It is called a ticksheet because users make selections by ticking (selecting) items.

The ticksheet developer uses the Ticksheet dialog box in EpiCenter Manager or Web Builder to construct these ticksheets.

Transaction Filters

Transaction filters allow Momentum ticksheet users to filter dimension demographics by behavior, such as “Show me how many people *called* (the behavior) Customer Support from Region Y in 1998.” Transaction filters are always associated with a fact table and transaction type, and a set of regular attribute filters. Optionally, transaction filters can be configured to apply a measure filter to the result of that filter set. See *Momentum*.

Transaction Type

Transaction types distinguish rows with different interpretations in the same fact table. For example, an Order fact table might contain both a BOOK and SHIP transaction type, differentiated by the value of the column *transtype_key*.

Measure terms created via EpiCenter Manager or Web Builder specify a transaction type in addition to a fact column. This allows the summation of only those rows in a fact table with the same transaction type.

GLOSSARY

Web Builder

Web Builder (**WebBuilder.exe**) is an Epiphany program for configuring user-interface metadata, including measures, ticksheets, and glossary entries. The Web Builder program, which is a subset of EpiCenter Manager, allows its users to configure ticksheets, without having access to the full-functionality of EpiCenter Manager.

INDEX

A

- A and B tables 50, 83
 - browsing aggregates 105, 113
 - most current 339
 - toggling 50, 221
- Access rights
 - group/user precedence 201
 - user/group precedence 205
- Acrobat PDF Files 16
- Actual tables 36, 103, 345
- Actuals metadata, Export All 326
- Adaptive Schema Generator 41, 305, 326, 345
- Add Column dialog box 207
- Add Job Steps dialog box 189
- ADD_DAYS SQL macro 270
- ADD_MONTHS SQL macro 270
- Administrative rights 202
- Administrator group 256
- Agg metadata 326
- AGG system call 258
- agg.exe. *See* Aggbuilder
- agg_timestamp, Aggbuilder log directory 58
- Aggbuilder 30, 35, 42, 54, 58, 102, 326, 345
 - browsing dimension 105
 - browsing fact 113
 - log file structure 58
 - program steps 59
- Aggregate building
 - extraction phase 42
 - invoking 58
- Aggregate Group dialog box 105, 115, 116
- Aggregate groups 115, 345
 - and dimension roles 116
 - default group 117
- Aggregate navigation 232, 346
- Aggregate operators 110
- Aggregate tables 62
- Aggregates
 - base dimension 347
 - Best & Worst 151
 - browsing dimension 105
 - browsing fact 113
 - building for Influence 160
 - fact 55
 - Influence 151
 - Lifecycles 151
 - optimal number of 105
 - Profiling 150
 - purpose of 55
 - Quarter Projections 150
 - Relevance 150
 - Trends 151
 - vs. custom fact indexes 111
- Aggregation 62
- AGGVERIFY system call 259
- all_groups_, MomentumBuilder table 62
- all_individuals_, MomentumBuilder table 62
- Allow Anonymous authentication 254
- API, native for EpiCenter 49
- Application Log Full Error 342
- Application Server 42, 346
 - aggregate navigation 232
 - aggregates at query time 59
 - aggregates rebuilt 233
- APPSERVER.LOG 343
- cannot log in 331–333
- config_master table 232
- EpiAppService NT Service 237
- error messages 330–343
- GIFs 338
- log files 228, 245
- log naming conventions 246
- memory for 334
- mx64M 334

INDEX

- proxy logging 243
- query machinery 356
- refresh time 235
- RefreshApp 236
- refreshing 232, 233, 234
- Registry keys 240
- restarting 233
- running from command line 231
- security 250
- security logs 333
- Security Manager log 248
- Service Control Manager 333
- setting up new domain 255
- SRV log 343
- starting from console 230
- starting/stopping Windows NT service 227
- time navigation 233
- verifying operation 228
- Windows NT authentication 231
- Windows Registry 232
- ApplicationServerPort Registry keys 240
- APPSERVER.LOG 333, 343
- AppServerHost Registry key 240
- APPSERVERHOST system call 259
- AppServerLogVerbosity Registry key 241
- APPSERVERPORT system call 259
- AppSessionTimeout Registry Key 241
- Arithmetic operators, RPN 121
- ASSERT_INDEX_EXISTS SQL macro 270
- Attribute dialog box 131, 133
- Attribute roles 346
 - assigning on ticketsheets 132
 - Relevance ticketsheets 146
- Attributes 25, 346
 - Clarity 130
 - modifying 133
 - Momentum 171
 - Relevance 130
 - steps for defining 133
 - ticketsheet 130
- Attributes dialog box 133
- Authentication
 - basic 254
 - modules 252
 - Pass through 253
 - Windows NT domain 252

B

- Backlog types 121, 347
- Base dimension aggregates 347
- Base Dimension dialog box 98, 99, 102, 105, 107, 114, 161, 168
 - selecting physical types 299
- Base dimension staging tables
 - populating 307
 - queries with joins 306
 - SQL statements 303
- Base dimension tables 23, 56, 98, 347
 - defining 99
- BASE HREF tag 338
- Base tables 56, 347
- Basic authentication 254
- BEGIN_ASSERT_INDEX SQL macro 270
- Best & Worst
 - aggregates 151
 - default ticksheet 147
- Bookings 319
- BOOL_TO_YN SQL macro 270
- Breakpoints, setting 79
- Build Registry key 241

C

- Calendars
 - 4-4-5 quarters 32, 217
 - fiscal quarter 32
- Can be used in Momentum filter 101
- CASE_BEGIN SQL macro 271
- CASE_ELSE SQL macro 271
- CASE_ELSEIF SQL macro 271
- CASE_END SQL macro 271
- CAT SQL macro 271
- CBIN_VAL SQL macro 272
- CHAR_1 SQL macro 272
- CHARTSLOGFILE system call 259
- ChartsOutputDir Registry Key 241
- CHARTSOUTPUTDIR system call 259
- chn_timestamp, EpiChannel log directory 58

- Choose EpiCenter dialog box 88, 92
- Choose Groups dialog box 130, 206
- Choose Measure dialog box 156
- Choose User dialog box 130
- Clarity 13, 20, 31, 124
 - Tables & Charts 124
 - ticksheet configuration 126–146
- Classification targets 158
- Classification tree, Influence 152
- Cleansing tools, used by external tables 50
- Clustered index, fact table 111
- Clusters 170
 - guidelines for 170
 - setting up on fact table 173
- Column sets 102, 116
- COLUMN_FILTER SQL macro 266
- COLUMN_LAST_VALUE SQL macro 266
- COLUMN_RANGE_FILTER SQL macro 266
- com.epiphany.security.EpiNTLogon security module 250
- com.epiphany.security.SecurityManager class 246
- com.epiphany.server.Server class 245
- Commands
 - epiappservice 237–240
 - extract.exe 186
 - jre 230, 334
 - MomentumBuilder.exe 60
 - refresh.exe 232–237
 - RefreshApp.exe 232, 236
 - Regedit.exe 329
 - SecMgr.exe 357
 - WebBuilder.exe 360
- config_master table 50, 232, 285
- Configuration dialog box 97, 119, 126, 162, 163, 216, 285–294, 339
 - Ticksheet Types 293
- Configuration, EpiCenter 97
- Constellations 21, 348
 - multiple 86
 - setting up 106–176
- Copy Ticksheet Items 145
- COUNT DISTINCT, SQL operator 120
- Count feature, Momentum 163
- Count Unjoined semantic type 52, 316
- COUNT, SQL operator 120
- COUNT_ROWS SQL macro 272
- COUNTER SQL macro 272
- Counts 170, 174
- CREATE_INDEX_IF_NOT_EXISTS SQL macro 272
- CURR SQL macro 265
- Currencies
 - multi-currency option 289
 - system usage 289
- Currency measurement 120
- current_datamart A / B 286
- current_datamart value 50, 233, 338
- current_datamart, EpiCenter Manager option 83
- Custom Fact Index semantic type 52, 63, 111, 317, 348
- Custom fact indexes 62–64, 111, 348

D

- Data is valid as of... 287
- Data merging extraction phase 42, 51
- Data Store dialog box 177, 179, 181
- Data stores 42, 43, 348
 - creating 177
 - EpiMart 43, 180
 - generic ODBC 179
 - JobFileLog 43, 180
 - LoggingDB 44, 180
 - Oracle 179, 262
 - SQL Server 179
- Data warehouses 15, 19, 20, 44
- Database administration tools 40
- Database server
 - registering 87–89
- DATABASE system call 259
- DatabaseName Registry key 241
- DatabasePassword Registry key 241
- DatabaseServer Registry key 241
- DatabaseType Registry key 241
- DatabaseUsername Registry key 241
- Dataset dialog box 127

INDEX

- Datasets 128, 213, 348
 - assigning ticketsheets to 127
 - ordering of 214
- Date base dimension tables 98
- Date Dimension dialog box 91, 117
- Date dimension fields 295
- Date dimension tables 24, 31, 117, 215, 349
 - populating 91, 214, 216
- Date range, EpiMart 216
- Date, displayed on reports 339
- Date.fy_name 205, 209
- DATE_FILTER SQL macro 267
- date_key 214, 309, 349
- DATE_LAST_VALUE SQL macro 267
- date_modified column 305
- DATE_RANGE_FILTER SQL macro 267
- date_type 286
- DBNOW SQL macro 273
- DBVENDOR system call 259
- DDL_BEGIN SQL macro 273
- DDL_BEGIN_NO_DECLARE SQL macro 273
- DDL_END SQL macro 274
- DDL_EXEC SQL macro 274
- Debug level, job 191
- DEBUG SQL macro 284
- DEBUG_LEVEL system call 259
- Debugging. *See* Scrutiny
- DECLARE_BEGIN SQL macro 275
- DECLARE_BODY SQL macro 275
- Defaults report folder
 - user and group 210
- Degenerate Dimension dialog box 108
- Degenerate dimensions 108, 348, 349
- Description Registry key 242
- Dialog boxes
 - Add Column 207
 - Add Job Steps 189
 - Aggregate Group 105, 115, 116
 - Attribute 131, 133
 - Attributes 133
 - Base Dimension 98, 99, 102, 105, 107, 114, 161, 168, 299
 - Choose EpiCenter 88, 92
 - Choose Groups 130, 206
 - Choose Measure 156
 - Choose User 130
 - Configuration 97, 119, 126, 162, 163, 216, 285–294, 339
 - Data Store 177, 179, 181
 - Datasets 127
 - Date Dimension 91, 117
 - Degenerate Dimension 108
 - Dimension Column 101, 172
 - Dimension Role 107, 162
 - Execute Job 191
 - Exporting Metadata 218
 - External Tables 299
 - Extractor 177, 178, 181
 - Extractor Steps 181, 182, 184, 188
 - Fact Column 110
 - Fact Table 108, 110, 111, 112, 113, 173
 - Filter 138, 176
 - Filter Elements 138
 - Filter Group 137
 - Filters 135
 - Find Report 211
 - Generate Schema 214
 - Glossary Entry 134, 142
 - Importing Metadata 219
 - Influence ticketsheet 155
 - Initialize EpiCenter 89, 90, 161
 - Job 189, 190, 191
 - Job Steps 192
 - Measure 118, 120, 122, 140, 157, 175
 - Measure Builder 118, 122, 123
 - Measure Column Element 141
 - Measure Sets 156
 - Measures 122
 - Momentum ticketsheet 171
 - New Constellations 161
 - Populates fact table option 307
 - Register EpiCenter 89
 - Report 95, 144, 212
 - Semantic Instance 188
 - Server Properties 88, 221, 222
 - SQL Query 139
 - SQL Statement 185, 186, 187, 304
 - SQL Statement, Tables References option 186, 304
 - SQL Statement, Template feature 187
 - Ticketsheet 126, 129, 134, 141, 144, 145, 146
 - Transaction Filter 174, 175
 - Truncate Tables before Extraction 199
 - User 205

- Dimension aggregates 59, 102, 105
 - defining 103
 - trade-offs 105
- Dimension Column dialog box 101, 172
- Dimension column sets 102
 - defining 103
 - multiple 105
- Dimension columns 349
 - displaying values of 208
 - restricting access to groups 203
 - restricting access to users 207
 - UNKNOWN 102
- Dimension Role dialog box 107, 162
- Dimension roles 23, 348
 - custom fact index 112
 - defining 107
 - in aggregate group 116
- Dimension semantic types 52, 311–316
- Dimension tables 20
 - defining dimension roles 107
 - hierarchical relationship of 25
 - number required 99
- Dimensions
 - attributes of 25
 - degenerate 108
- DimRoleName _sskey 309
- DIRNAME system call 259
- Domains 254
- Driver Registry key 243
- DROP_INDEX SQL macro 275
- DROP_TABLE_IF_EXISTS SQL macro 275
- DSN Registry key 242
- DSN system call 260
- Duplication feature, EpiCenter Manager 94

E

- Elements, measure column 118
- ELSE SQL macro 275
- E-mail
 - configuring for job status 196
 - EpiCenter Manager default address 198, 285
 - password 285
- End of Extraction 50, 58, 183
- End of Extraction job 339
- END_ASSERT_INDEX SQL macro 275
- END_IF SQL macro 276
- end_year 286
- EOS SQL macro 276
- EpiAdminServer 248
- epiappservice command 237–240
- EpiAppService. *See also* epiappservice 237
- EPIBIN system call macro 54, 61, 260
- EpiCenter 13, 21, 44
 - current configuration 97
 - initializing 89–92
- EpiCenter Manager 30, 39, 86, 94, 233, 234, 353
 - cloning tables 94
 - configuration 95
 - conventions 94
 - defining base dimension tables 99
 - duplication feature 94
 - Export all command 326
 - Generate Schema 214
 - invoking commands 93
 - measure terms 359
 - registering server 87–89
 - starting the program 87
 - updating data 94
 - window 93
- EpiCenter Manager Report 95
- EpiCenter Manager tables
 - _A and _B sets 106
- EpiCenters 39
 - appropriate number of 86
 - backing up definitions 323
 - producing new 219
 - registering existing 92
 - setting up 97
 - unregistering 94
- EpiChannel 30, 40, 41, 43, 45, 54, 58, 257
 - Also see* extract.exe
 - error messages 78
 - logging 81
 - memory 48
 - output 75, 80
 - selecting debug level 186
 - trial-run model 79
 - verbose debugging option 78
 - verbosity levels 191

INDEX

- EpiChannel log file 314, 315, 317
 - sskey rows 312, 316
- EpiChannel output 75
- EPIINT SQL macro 276
- EPIKEY SQL macro 276
- EpiLoginException violations 230
- EpiMart 21, 30, 42
 - base tables 56
 - data range 216
 - data store 43, 180
 - initial load of 315
 - schema changes in 35
 - tables 39, 83
- EpiMart dimension column values displaying 208
- EpiMart tables 30
 - building 35
 - mirroring of 106
 - naming conventions 35
 - purging 199
 - testing with Scrutiny 223
 - toggling A and B 50
- EpiMeta 21, 28, 29, 35, 39, 323
 - cloning 326
 - database table schema 85
 - tables 29, 324
- EpiNTLogon 255
 - Windows NT authentication module 250
- EpiPassThruLogon module 333
- Epiphany
 - database schema 19
 - Release 3.4 features 14
 - setting up system users 205
 - system overview 27–29
- Epiphany Application Server. *See* Application Server
- Epiphany Application Suite 31
- Epiphany Customer Support 329
- Epiphany macros
 - SQL 262–284
 - system call macros 257–261
 - translation of 265
- Epiphany software authentication modules 252
- Epiphany vendor-independent macros 264
- Epiphany.dll proxy 225, 243
- Error messages
 - Application Log Full 342
 - Application Server 330–343
 - Cannot Connect to the Server 330
 - EpiChannel 78
 - Object Not Found 340
 - out of memory 334
 - Refresh program fails 342
 - Registry Editor warning 329
 - SQL Server 330
 - User Cannot Log In 331
 - VirtualMartDatabase Key Missing 334
- EXC system call 260
- EXC_ARGS system call 260
- EXC_CMD system call 260
- EXCVERIFY system call 260
- Execute Job dialog box 191
- Export All command 326
- Export machinery 324
- Export tables
 - Export_col 327
 - Export_rel 327
 - Export_row 327
 - Export_status 327
 - Export_tbl 327
 - Rel_parent 327
- export.mdb 219
- Export_col table 327
- Export_rel table 327
- Export_row table 327
- Export_status table 327
- Export_tbl table 327
- Exporting Metadata dialog box 218
- Exporting ticksheets 324
- External tables 41, 50, 194
 - cleansing tools 50
 - input to staging queries 310
 - last_extract_date 50, 194
 - truncation of 198
- External Tables dialog box
 - selecting physical types 299
- extract.exe
 - Execute button in EpiCenter Manager 192
- extract.exe command 186
 - Also see* EpiChannel
 - directory location 54

Extraction
 aggregate building phases 42
 data merging phase 42, 51
 defining groups 184
 extraction steps for groups 184
 filter macros 182
 groups 182
 load phase 41, 51
 multi-stage 53
 setting up for EpiCenter 177
 stages 41
 steps 184
 Extraction Date Unknown 339
 Extraction groups 46
 Extraction statements 50
 Extraction step 46
 Extractor dialog box 177, 178, 181
 Extractor filters 180
 Extractor steps 46
 Extractor Steps dialog box 181, 182, 184, 188
 Extractors 41, 46, 47
 adding as job step 193
 defining 181
 End of Extraction 50, 58
 Extractor Steps dialog box 182
 shared by jobs 47

F

Fact aggregates 55, 59
 browsing 113
 Fact Column dialog box 110
 Fact columns
 defining 110
 queyable 356
 Queyable option 110
 Fact semantic types 52, 316–322
 Fact staging tables 64
 loading 322
 Fact Table dialog box 108, 110, 111, 112, 113, 173
 Fact tables 20, 25, 34
 clustered index 111
 defining 108–109
 reloading populated 321

FACTMONEY 276
 FACTMONEY SQL macro 276
 FACTQTY SQL macro 276
 Facts
 in EpiMart 25
 rows with zero facts 311
 SQL limits 84
 FILENAME system call 260
 Filter dialog box 138, 176
 Filter elements 136
 defining 138–140
 Filter Elements dialog box 138
 Filter Group dialog box 137
 Filter groups 136
 defining 137–138, 138–140
 Filter Households ticksheet type 171, 293
 Filter Individual Momentum ticksheet 171
 Filter Individual ticksheet type 171, 293
 filter_block_key column 323
 FilterGroupCols, Momentum role type 171
 FilterIndividualCols, Momentum role type 171
 Filters
 defining 134–140
 Momentum ticksheet 172
 ticksheet 134
 Filters dialog box 135
 Find Report dialog box 211
 First Dimension Value semantic type 314
 Flat files, source system 43
 FLOAT SQL macro 276
 Foreign keys 24, 31
 to date dimension 31

G

Generate Schema command 35, 36, 214
 Generate Schema dialog box 214
 GIFs, generated by Application Server 338
 Glossary entries 213
 adding filter group 138
 ticksheet items 134
 Glossary Entry dialog box 134, 142

INDEX

Group by flags, selecting 104
GROUP BY statements 306
GROUP BY, SQL clause 104
Groups
 assigning ticksheet access to 129
 setting up 201–205

H

Household, Momentum ticksheet type 125
HOUSEHOLD_CARDINALITY 291

I

IDENTITY SQL macro 276
IF SQL macro 276
IIS security 254
IIS Web server. *See* Web server
IJ_FROM SQL macros 277
ikey 318
Import machinery 324, 325
Importing Metadata dialog box 219
Ind_Group_Joiner fact table 164
index.pdx, Acrobat full-text index 16
Indexed for fast Clarity and Relevance search option 101
Indexing 62
 custom fact 111
 fact table clustered index 111
 leading term 111
Individual, Momentum ticksheet type 125
INDIVIDUAL_CARDINALITY 291
Influence 125
 aggregates 151, 160
 classification tree 152
 determining primary dimension 154
 determining source attributes 154
 determining the target 154
 measure sets 154, 156
 Momentum lists as attributes 159

 regression tree 152, 153
 tips 159
Influence ticksheet dialog box 155
Influence ticksheets
 configuring 152
 primary dimension 153
 setting up 153–158
 source attributes 153
 target 153
Initial Load Dimension semantic type 52, 313, 315
Initial Load Fact semantic type 52, 321
INITIAL_LOAD SQL macro 269
Initialize EpiCenter dialog box 89, 90, 161
INSTANCE_NAME system call 260
InstanceRootDir Registry key 242
INSTANCEROOTDIR system call 260
INSTR SQL macro 277
Inventory data, transactional system 34
ISQL 339
iss sskey 312
ISS system call 260

J

Job data store roles 54
Job dialog box 189, 190, 191
Job steps 46
 ordering 189
Job Steps dialog box 192
JOB_NAME system call 261
JobFileLog data store 43, 180
JobLogFile 179
Jobs 75
 debug level for 191
 default 189
 disabling 190
 e-mail notice of status 196
 monitoring 81–82
 role of 45
JOIN_LEFT_OUTER SQL macro 277
JOIN_RIGHT_OUTER SQL macro 277
JOIN_WHERE SQL macro 278
jre command 230, 334

L

last_extract_date 50, 194, 287, 339
 Latest Dimension Value semantic type 313
 Leading terms, fact table indexing 111, 348
 Leaf dimensions 165, 167
 LENGTH SQL macro 278
 Lifecycles 125
 aggregates 151
 default ticksheet 149
 Listboxes
 dynamic 136
 filter block 136
 Load extraction phase 41, 51
 Log database 58, 81
 Log device 81
 Logging Data Store 179
 Logging, proxy 243
 logging_mssql.sql script 179
 LoggingDB data store 44, 180
 Logs
 location of 80
 role of 80
 LOJ_FROM SQL macro 278

M

Macros
 Ephipany vendor-independent 264
 semantic instance 46
 SQL 49
 Mail Profile name 198, 285
 MARTDBNAME SQL macro 269
 MAX_SYS_DATE SQL macro 278
 Measure Builder dialog box 118, 122, 123
 Measure Column Element dialog box 141
 Measure column elements 118
 Measure dialog box 118, 120, 122, 140, 157, 175
 Measure mapping
 defined 118
 verifying 143
 Measure sets
 Influence 154, 156
 Measure Sets dialog box 156
 Measure terms 120, 359
 Measure units 119, 289
 defining 289
 symbols for 289
 Measurement units 120
 Measures 118, 348
 defining for a constellation 118
 mapping elements to 143
 Measures dialog box 122
 Metadata 21
 current state of 215
 exporting individual objects 218
 exporting/importing of 217
 overwriting 325
 primary keys in tables 323
 ticksheet 324
 METADBNAME SQL macro 269
 Microsoft Access
 database 218, 221
 database tables 324
 Export database 326
 Microsoft Internet Explorer 342
 Microsoft Outlook Exchange
 EpiChannel e-mail 197
 Microsoft SQL Server
 cannot connect to 330
 error messages 330
 Physical type values 299
 TCP/IP sockets 336
 Microsoft SQL Server data stores 179
 min_sample_invlog10 286
 Mini-dimensions 168–169
 MODULO SQL macro 278
 mom_inlist_limit 287
 Momentum 14, 20, 31, 125, 160, 291
 Can be used in filter option 101
 clusters 170, 173
 constellations 161, 165
 count feature 163
 counts 170, 174
 defining transaction filters 174–176
 Filter Household ticksheet type 171
 Filter Households ticksheet type 293

INDEX

- Filter Individual 171
- Filter Individual ticksheet type 293
- FilterGroupCols role type 171
- FilterIndividualCols role type 171
- HOUSEHOLD_CARDINALITY 291
- INDIVIDUAL_CARDINALITY 291
- leaf dimensions 167
- mini-dimensions 168–169
- modifying default labels 291
- MOMENTUM_HOUSEHOLD_LABELS 291
- MOMENTUM_INDIVIDUAL_LABELS 292
- MOMENTUM_MEASURE_MODE 292
- Option labels for ticksheets 162
- order of extraction for 60
- queryable fact column 356
- saving filter settings 176
- setting up attributes 171
- slowly changing dimensions 165
- testing with Scrutiny 223
- ticksheet filters 172
- transaction filters 169
- Momentum Adjacent constellations 86, 166, 348
 - setting up 166
- Momentum constellations 348
- Momentum lists
 - tips for using with Influence 159
 - using as Influence attributes 159
- Momentum sampling tables 214, 215
- Momentum Ticksheet dialog box 171
- Momentum ticksheet types 125
- MOMENTUM_HOUSEHOLD_LABELS 291
- MOMENTUM_INDIVIDUAL_LABELS 292
- MOMENTUM_MEASURE_MODE 292
- MomentumBuilder 60
 - all_groups_table 62
 - all_individuals_table 62
 - default job 61
 - Java file for 61
 - system call 189
 - system call job step 195
 - verifying extraction 62
- MomentumBuilder.exe 60
 - command-line arguments 60–61
- Monitors 190
- MSSQL Server SQL Service 233
- Multi-currency option 289
- mx64M parameter, Application Server 334

N

- NBIN_VAL SQL macro 279
- Netscape Navigator 342
- New Constellations dialog box 161
- NEXT SQL macro 265
- No Results Available for a Query 338
- NO_FROM_LIST SQL macro 279
- NOT_DEBUG SQL macro 284
- NOT_INITIAL_LOAD SQL macro 269
- NT. *See* Windows NT
- NTLM authentication 254
- NUMBER() SQL macros 279
- number_of_years 286
- NVL SQL macro 279

O

- Object Not Found 340
- ODBC
 - data store 179
 - layer 43
 - source system 43
 - SQL driver 330
- fiscal_year start_m 286
- Option labels
 - modifying default Momentum 291
 - ticksheet 290
- Options Labels
 - Momentum ticksheet 162
- Oracle
 - data store 179
 - Physical type values 299
 - SQL macros 262, 279
 - tablespaces 262, 263
- ORACLE SQL macro 279
- Out of Memory exception error 334
- Output Data Store 178

P

Pass-through authentication 253
 PASSWORD system call 261
 PATH system call 261
 PDF files 16
 full-text index 16
 Per cent measurement 120
 permission 249
 Permissions
 default for new users 205
 effect of changing on Application Server 234
 Security Manager 222
 setting for ,viewing folder/report 212
 ticket sheet 249
 Physical type values
 Oracle 299
 SQL Server 299
 Physical types 299
 selecting for fact column 110
 Pipelined semantic type 320
 Populates a dimension option 304
 Populates fact table option 307
 Postfix 289
 Preserve existing items option 145
 Primary dimension
 determining 154
 Influence ticket sheet 153
 Primary keys 31, 323
 correspondence to sskeys 305
 process_key 309, 318, 319, 320, 321
 Profiling
 aggregates for 150
 default ticket sheet 147
 Relevance Profiling 125
 PROGRAM_NAME system call 261
 Propagate button 288
 proxy.log file 242
 ProxyLogFile Registry key 242
 Public Defaults folder 210
 Pull/push SQL statements 356

Q

QM_randomly_generated_alphabetic_sequence_use
 name 333
 Quarter Projections 125
 aggregates 150
 default ticket sheet 149
 Query machinery 42, 233, 356
 Query performance, improving 62
 Query run time
 setting for user 206
 Queryable fact column, Momentum 356
 Queryable option, fact column 110

R

RAISE_EXCEPTION SQL macro 280
 RDBMS 27
 indexes 32
 physical types for platform 299
 server 43
 Refresh system call 189
 refresh.exe command 232–237, 334
 fails 342
 RefreshApp.exe command 232, 236
 Regedit.exe command 329
 Register EpiCenter dialog box 89
 Registry Editor warning 329
 Registry keys
 Application Server 240
 ApplicationServerPort 240
 AppServerHost 240
 AppServerLogVerbosity 241
 AppSessionTimeout 241
 Build 241
 ChartsOutputDir 241
 DatabaseName 241
 DatabasePassword 241
 DatabaseServer 241
 DatabaseType 241
 DatabaseUsername 241
 Description 242
 Driver 243
 DSN 242

INDEX

- InstanceRootDir 242
- ProxyLogFile 242
- SecurityClass 243, 250, 253
- SystemLogDir 242
- SystemLogDirWebpath 242
- SystemLogWebDir 249
- TempDirGarbageLifetime 242
- TemplateDir 242
- REGISTRY_EIPATH system call 261
- REGISTRY_ROOT system call 261
- Regression targets 158
- Regression tree, Influence 152, 153
- Rel_parent table 327
- Relational database management source systems. *See* RDBMS
- Relevance 13, 20, 31, 124
 - aggregates for 150
 - Best & Worst 124, 147
 - date dimension for trends 216
 - default ticksheets 147
 - Influence 125
 - Lifecycles 125
 - Profiling 147
 - Quarter Projections 125, 149
 - ticksheet attribute roles 146
 - ticksheet configuration 126–158
 - Trends 125, 148
- Relevance ticksheets 356
- Reload Max Date semantic type 322
- RENAME_OBJECT SQL macro 280
- Report dialog box 95, 144, 212
- Report Gallery 144, 200, 209, 357
- Reports 357
 - copying/moving 211
 - date displayed on 339
 - folders for saved 211
 - saved for ticksheet 144
- Result Page Error
 - Extraction Date Unknown 339
- Reverse Polish Notation 120, 122–124
 - arithmetic operators 121
 - translation of 122
- RPN. *See* Reverse Polish Notation
- **S**

- SAM 254, 255
- Sampling tables, Momentum 214, 215
- Save and Restore Manager 249
- SAVE_RESTORE 333
- Schema 345
 - changes in Epimart 35
 - definitions 36
 - generating 214
 - operations 37
- Scrutiny debugging tool 223, 357
 - testing Application Server 227
- SecMgr.exe 222
- SecMgr.exe command 222, 357
 - See also* Security Manager
- Security
 - access rights 200
 - authentication 200
 - Epiphany system 200
 - Web browser caching 199
 - Windows NT groups 200
- Security Access Monitor. *See* SAM
- Security logs
 - Application server 333
 - APPSERVER.LOG 333
 - QM_randomly_generated_alphabetic_sequence... 333
 - SAVE_RESTORE 333
- Security Manager 87, 222, 234, 250, 357
- SecurityClass Registry key 243, 250, 253
- SecurityManager log 246, 248
- SELECT DISTINCT query 307
- SELECT statements 307
- SELECT_INTO_BEGIN SQL macro 280
- SELECT_INTO_BODY SQL macro 280
- Selection column 357
- Semantic Instance dialog box 188
- Semantic instances 35, 40, 42, 46, 49
 - defined 51
 - setting up 188–189
- Semantic templates 305
 - defined 51
- Semantic transformation 51

- Semantic types
 - Count Unjoined 52, 316
 - Custom Fact Index 52, 63, 111, 317, 348
 - defined 51
 - defining 52
 - dimension 52
 - fact 52
 - First Dimension Value 314
 - Initial Load Dimension 52, 313, 315
 - Initial Load Fact 52, 321
 - Latest Dimension Value 313
 - Pipelined 320
 - Reload Max Date 322
 - Slowly Changing Dimensions 311, 315
 - Transactional 317
 - transactional 317
 - Transactional/State-like 318, 322
 - Transactional/State-like/Force Close 319
- Semantic types base dimension and fact tables 52
- Semantics 39, 40
- Server Properties dialog box 88, 221, 222
- SERVER system call 261
- Servers
 - unregistering 94
- Service Control Manager
 - fails to start 333
- Shipments 319
- Slowly Changing Dimensions 315
- Slowly changing dimensions 313
 - Influence 160
 - Momentum 165
 - semantic type 311
- Slowly changing dimensions semantic type 315
- SMALLDATE SQL macro 280
- SMALLINT SQL macro 280
- Source attributes
 - Influence tick sheets 153, 154
- Source system identifier keys. *See* sskeys
- SQL
 - dialect problems 186
 - limits for facts 84
 - sample for populating a table 187
- SQL macros 49, 257, 268, 276, 305
 - ADD_DAYS 270
 - ADD_MONTHS 270
 - ASSERT_INDEX_EXISTS 270
 - BEGIN_ASSERT_INDEX 270
 - BOOL_TO_YN 270
 - CASE_BEGIN 271
 - CASE_ELSE 271
 - CASE_ELSEIF 271
 - CASE_END 271
 - CAT 271
 - CBIN_VAL 272
 - CHAR_1 272
 - COLUMN_FILTER 266
 - COLUMN_LAST_VALUE 266
 - COLUMN_RANGE_FILTER 266
 - COUNT_ROWS 272
 - COUNTER 272
 - CREATE_INDEX_IF_NOT_EXISTS 272
 - CURR 265
 - DATE_FILTER 267
 - DATE_LAST_VALUE 267
 - DATE_RANGE_FILTER 267
 - DBNOW 273
 - DDL_BEGIN 273
 - DDL_BEGIN_NO_DECLARE 273
 - DDL_END 274
 - DDL_EXEC 274
 - DEBUG 284
 - DECLARE_BEGIN 275
 - DECLARE_BODY 275
 - DROP_INDEX 275
 - DROP_TABLE_IF_EXISTS 275
 - ELSE 275
 - END_ASSERT_INDEX 275
 - END_IF 276
 - EOS 276
 - EPIINT 276
 - EPIKEY 276
 - Epiphany 262–284
 - FACTQTY 276
 - FLOAT 276
 - IDENTITY 276
 - IF 276
 - IJ_FROM 277
 - INITIAL_LOAD 269
 - INSTR 277
 - JOIN_LEFT_OUTER 277
 - JOIN_RIGHT_OUTER 277
 - JOIN_WHERE 278
 - LENGTH 278
 - LOJ_FROM 278
 - MARTDBNAME 269
 - MAX_SYS_DATE 278
 - METADBNAME 269

INDEX

- MODULO 278
- NBIN_VAL 279
- NEXT 265
- NO_FROM_LIST 279
- NOT_DEBUG 284
- NOT_INITIAL_LOAD 269
- NUMBER() 279
- NVL 279
- ORACLE 279
 - Oracle-specific 262
- RAISE_EXCEPTION 280
- RENAME_OBJECT 280
- SELECT_INTO_BEGIN 280
- SELECT_INTO_BODY 280
- SMALLDATE 280
- SMALLINT 280
- SQLSERVER 280
- SSKEY 281
- SUBSTRING 281
- SUPERNVL 281
- TABLE_EXISTS_CONDITION 281
- TABLE_WITH_PREFIX 281
- TIMESTAMP_FILTER 268
- TIMESTAMP_FILTER_RANGE 268
- TINYINT 281
- TO_CHAR 281
- TO_DATE 282
- TO_DATEFMT 282
- TO_EPIDATE 282
- TO_NUMBER 282
- TO_TIME 282
- TO_YYYYMMDD 282
- TRANSLATE 283
 - usage 264
- VAR 283
- VAR_ASSIGN_BEGIN 283
- VAR_ASSIGN_END 283
- VAR_ASSIGN_INTO 283
- VARCHAR 283
- YYYYMMDD_FILTER 269
- YYYYMMDD_FILTER_RANGE 269
- YYYYMMDD_LAST_VALUE 269
- SQL operations 46
- SQL operators 120
 - COUNT 120
 - COUNT DISTINCT 120
 - SUM 120
- SQL Query dialog box 139
- SQL SELECT statements
 - fill from Query 139
- SQL Server. *See* Microsoft SQL Server
- SQL Statement dialog box 185, 186, 187
 - re-sizing 187
 - sample SQL 187
 - Tables References option 186, 304
 - Template feature 187, 304
 - zooming 187
- SQL statement extraction step 48
- SQL statements 46
 - error in 49
 - fact staging 307
 - populating staging tables 186, 303
 - pull/push 356
 - push/pull 48
 - setting breakpoints 79
 - stand-alone 48, 359
- SQLNET system call 261
- SQLSERVER SQL macro 280
- SRV log 343
- ss_keys 309
- SSKEY SQL macro 281
- sskeys 64, 65, 178, 305, 309, 312, 314, 315, 317, 318, 319, 322
 - duplicate 306
 - entering a pipeline stage 320
 - ikey 312, 318
 - new rows 312
 - UNKNOWN row 65
- Staging tables 39, 41, 49
 - truncating 100, 191, 198
- Star schemas 20, 39, 99, 306
- start_day_445 287
- start_year 287
- StorageManager log 246
- SUBSTRING SQL macro 281
- SUM, SQL operator 120
- SUPERNVL SQL macro 281
- SVGA monitors 190
- Symbol character, measure unit 289
- Synchronized groups 255
- System calls 53, 359
 - adding as job step 193
 - AGG 258
 - AGGVERIFY 259

APPSERVERHOST 259
 APPSERVERPORT 259
 CHARTSLOGFILE 259
 CHARTSOUTPUTDIR 259
 DATABASE 259
 DBVENDOR 259
 DEBUG_LEVEL 259
 DIRNAME 259
 DSN 260
 EPIBIN 260
 EXC 260
 EXC_ARGS 260
 EXC_CMD 260
 EXCVERIFY 260
 exit code 54
 FILENAME 260
 INSTANCE_NAME 260
 INSTANCEROOTDIR 260
 ISS 260
 JOB_NAME 261
 macro syntax 258
 Momentum 189
 PASSWORD 261
 PATH 261
 PROGRAM_NAME 261
 Refresh 189
 REGISTRY_EPIPATH 261
 REGISTRY_ROOT 261
 SERVER 261
 SQLNET 261
 USER 261
 VERSION 261
 SystemLogDir Registry key 242, 247
 SystemLogDirWebpath Registry key 242
 SystemLogWebDir Registry key 249

T

TABLE_EXISTS_CONDITION SQL macro 281
 TABLE_WITH_PREFIX SQL macro 281
 Tables
 base 347
 metadata 323
 rebuilding all 215
 viewing updated 215
 Tables, A and B. *See* A and B tables

Tablespaces, Oracle 262
 alternate names for 263
 Target
 determining for Influence ticksheet 154
 Influence ticksheet 153
 TCP/IP sockets, SQL Server 336
 TempDirGarbageLifetime Registry key 242
 Template feature, SQL Statement dialog box 187
 TemplateDir Registry key 242
 TemplateServer 248
 Ticksheet access
 assigning 129
 assigning to groups 203
 assigning to users 207
 Ticksheet dialog box 126, 129, 134, 141, 144, 145, 146
 Cancel button 130
 Ticksheet type label 293
 Ticksheet types 293
 Momentum Filter Households 171
 Momentum Filter Individual 171
 Ticksheets 14, 27
 copying items 145
 customizing labels 290
 exporting 324
 filters 134
 glossary entries 134
 grouped by dataset 213
 metadata 324
 modifying 176
 option labels 290
 saved reports 144
 setting up attributes for 130
 types 124
 Time dimension (Date_0) 31
 Time navigation 233
 Time zone, displayed on reports 339
 Time, uniform treatment of 31
 TIMESTAMP_FILTER SQL macro 268
 TIMESTAMP_FILTER_RANGE SQL macro 268
 TIMESTAMP_LAST_VALUE 268
 TIMESTAMP_LAST_VALUE SQL macro 268
 Timestamps 71, 72, 74, 80
 ignore option 72, 191
 TINYINT SQL macro 281
 TO_CHAR SQL macro 281

INDEX

TO_DATE SQL macro 282
TO_DATEFMT SQL macro 282
TO_EPIDATE SQL macro 282
TO_NUMBER SQL macro 282
TO_TIME SQL macro 282
TO_YYYYMMDD SQL macro 282
Toolbar icons, EpiCenter Manager 93
Toplevel template 253
Transaction Filter dialog box 174, 175
Transaction filters 359
 defining Momentum 174–176
 Momentum 169
Transaction type dimensions 98
Transaction types 309, 359
 EpiCenter 288
 multiple 108
Transactional data, uniform treatment of 32
Transactional semantic types 317
Transactional/State-like semantic type 318, 322
Transactional/State-like/Force Close semantic type 319
Transactions, tracking changes in inventory 33
TRANSLATE SQL macros 283
Transtype base dimensions 98
Transtype dimensions. *See* Transaction type dimensions
transtype_0 table 288
transtype_key 309, 320
Trends 125
 aggregates 151
 default ticksheet 148
Truncate staging tables 100, 191
Truncate Tables before Extraction dialog box 199
Truncation
 external tables 198
 staging tables 198

U

Units measurement 120
UNKNOWN
 dimension column 102
 rows 65, 223
 sskey 313
 string 64, 65
Unregistering servers/EpiCenters 94
Update button 288
Update Unjoined. *See* Count Unjoined semantic type 316
User dialog box 205
USER system call 261
Users
 assigning ticksheet access to 129
 setting up 205

V

VAR SQL macro 283
VAR_ASSIGN_BEGIN SQL macro 283
VAR_ASSIGN_END SQL macro 283
VAR_ASSIGN_INTO SQL macro 283
VARCHAR SQL macros 283
Verbose option, EpiChannel 78
Version number, EpiCenter Manager 287
VERSION system call 261
VGA monitors 190

W

Web 249
 attributes as link to 132
 browser security considerations 199
Web browser 13, 27, 210
 login dialog box 254
Web Builder 87, 221, 233, 353
 measure terms 359

INDEX

Web server 27, 225, 243, 245, 338, 340, 346
 alias 249
 authentication 250
 authentication method 250
 finding alisas 338
 instance_name alias 337
 log file location 242
 Object Not Found 340
 starting 239
 stopping 239
 URL accessing 340
 URL address 225
Web server error message, Object Not Found 340
Web URL address 225
WebBuilder.exe command 360
week_start_day 287
Windows NT
 authentication 206, 231
 authentication module 250
 Authentication modules tips 253
 domain authentication 252
 group access rights 200
 group member synchronization 251
 importing users into groups 202
 Performance Monitor 81–82
Working directory 58, 80
 location of 80
World Wide Web (WWW). *See* Web.

Y

Years, beginning/end of EpiMart 216
Yen sign 289
YYYYMMDD_FILTER SQL macro 269
YYYYMMDD_FILTER_RANGE SQL macro 269
YYYYMMDD_LAST_VALUE SQL macro 269

INDEX